

Linear-Time FPT Algorithms via Half-Integral Non-returning A -path Packing

Yoichi Iwata
National Institute of Informatics
yiwata@nii.ac.jp

Yutaro Yamaguchi*
Osaka University
yutaro_yamaguchi@ist.osaka-u.ac.jp

Yuichi Yoshida†
National Institute of Informatics
yyoshida@nii.ac.jp

Abstract

A recent trend in the design of FPT algorithms is exploiting half-integrality of LP relaxations. That is, starting with a half-integral optimal solution to an LP relaxation, we assign integral values to variables one by one by branch and bound. This technique is general and the resulting time complexity has a small dependency on the parameter. However, the time complexity often becomes a large polynomial in the input size because we need to compute half-integral optimal LP solutions.

In this paper, we address this issue by observing that, for many problems, the LP relaxation can be seen as the dual of the LP relaxation of a non-returning A -path packing problem. Then, we provide an $O(km)$ -time algorithm that computes a half-integral optimal solution to the latter problem, where k is the optimal value and m is the number of edges. This improves the computational time even for a special case of the non-returning A -path packing, called internally-disjoint A -path packing. As a corollary, we obtain FPT algorithms for various problems, including GROUP FEEDBACK VERTEX SET, SUBSET FEEDBACK VERTEX SET, NODE MULTIWAY CUT, NODE UNIQUE LABEL COVER, and NON-MONOCROMATIC CYCLE TRANSVERSAL. For each of these problems, the obtained running time is linear in the input size and has the current smallest dependency on the parameter. In particular, these are the first linear-time FPT algorithms for GROUP FEEDBACK VERTEX SET and NON-MONOCROMATIC CYCLE TRANSVERSAL.

*Supported by JSPS KAKENHI Grant Number JP16H06931

†Supported by JST ERATO Kwarabayashi Large Graph Project

1 Introduction

1.1 FPT Algorithms using Half-Integral LPs

Parameterized complexity is a subject of studying the complexity of parameterized problems. We say that a parameterized problem with a parameter k is *fixed parameter tractable (FPT)* if we can solve the problem in $f(k)\text{poly}(n)$ time, where n is the input size. A variety of parameterized problems is known to be FPT. For the comprehensive list of FPT problems, see, e.g., [8, 10] and references therein.

One of the motivations to study parameterized complexity is understanding tractable subclasses of (NP-)hard problems, and hence the primary interest has been which parameterized problems admit FPT. However, from a practical point of view, it is crucial that the running time with respect to the input size be also small. Indeed, linear-time FPT algorithms, i.e., FPT algorithms whose running times are linear in the input size, are proposed for several problems including TREewidth [4], ALMOST 2-SAT [17, 32], FEEDBACK VERTEX SET (FVS) [3], SUBSET FVS [20], DIRECTED FVS [21], and NODE UNIQUE LABEL COVER [22]. Those works focused on reducing the running time with respect to the input size, and the dependency on the parameter is often sub-optimal. For example, SUBSET FVS admits an FPT algorithm with time complexity $O^*(4^k)$ ¹ [18] whereas the best linear-FPT algorithm has time complexity $O(25.6^k m)$ [20], where m is the number of edges in the input graph.

Recently, half-integrality of LP relaxations have been used to design FPT algorithms for a broad range of problems [9, 15, 18, 19, 35]. To see the idea, consider a minimization problem whose goal is finding a solution of size k , and suppose that it admits a half-integral LP relaxation². The algorithm is based on the standard branch-and-bound framework as follows. First, we compute a half-integral LP solution. If all the variables have integral values or the sum of the values in the LP solution exceeds k , we can stop. Otherwise, we fix variables with values 1. Then, we pick an arbitrary variable with value $1/2$, and we branch into the case that its value is fixed to 0 and the case that its value is fixed to 1. This approach has several big advantages: It can be applicable to a variety of problems just by changing the LP, and moreover, it has a small dependency on the parameter k . Indeed, for many problems including ALMOST 2-SAT [19], NODE MULTIWAY CUT [9], and GROUP FVS [18], the current smallest dependency on the parameter is achieved by this approach.

A drawback of the above-mentioned approach is that it is not trivial how to efficiently compute half-integral LP solutions. Iwata, Wahlström, and Yoshida [18] viewed half-integrality as a discrete relaxation and showed that one can compute half-integral LP solutions for ALMOST 2-SAT and (EDGE) UNIQUE LABEL COVER in time linear in the input size by reducing them to the s - t cut problem. Moreover, they showed that we can compute LP solutions with some extremal condition, which we call the *farthest condition* in this paper (see Section 4.2 for details), in the same running time. Using farthest solutions, the LP lower bound strictly increases for each branching. As a result, they obtained linear-time FPT algorithms for these problems.

¹ $O^*(\cdot)$ hides a polynomial dependency on the input size. When focusing on reducing the $f(k)$ part, the $\text{poly}(n)$ part is often ignored by using this notation.

² Most of the LPs used in FPT algorithms are not natural LP relaxations of the original problems but LP relaxations of *rooted* problems. For example, the rooted version of FVS is a problem of finding a minimum vertex subset S such that the graph obtained by removing S contains no cycles reachable from a prescribed vertex s . Note that the existence of a half-integral LP relaxation to the rooted problem does not imply 2-approximability of the original problem.

Table 1: Summary of our FPT results. Here, T_Γ denotes the time complexity for performing group operations and Σ is the alphabet, and m denotes the number of edges/constraints in the input. All the algorithms are deterministic excepting the $O(25.6^k m)$ -time algorithm for SUBSET FVS. See Appendix A for the problem definitions.

Problem	Smallest $f(k)$	Existing linear-time FPT	Our result
GROUP FVS	$O^*(4^k)$ [18]	—	$O(4^k km T_\Gamma)$
SUBSET FVS	$O^*(4^k)$ [18]	$O(25.6^k m)$ (randomized) [20] $2^{O(k \log k)} m$ (deterministic) [20]	$O(4^k km)$
NODE MULTIWAY CUT	$O^*(2^k)$ [9]	$O(4^k m)$ [5, 17]	$O(2^k km)$
NODE UNIQUE LABEL COVER	$O^*(\Sigma ^{2k})$ [18]	$ \Sigma ^{O(k \Sigma)} m$ [22]	$O(\Sigma ^{2k} km)$
NON-MONOCROMATIC CYCLE TRANSVERSAL	$O^*(4^k)$ [35]	—	$O(4^k km)$

Unfortunately, the s - t cut approach does not work well for other problems such as GROUP FVS, SUBSET FVS, NODE MULTIWAY CUT, and NODE UNIQUE LABEL COVER because this approach is essentially applicable only to edge-deletion problems³ and because the size of the auxiliary network is not linear in the input size in general, e.g., for SUBSET FEEDBACK EDGE SET, the size of the network becomes $2^{O(m)}$. For those problems, we need to resort to solving linear programs, and the resulting FPT algorithms have large dependencies on the input size. Among those problems, linear-time FPT algorithms have been obtained for NODE MULTIWAY CUT [5, 17], SUBSET FVS [20], and NODE UNIQUE LABEL COVER [22] by problem-specific arguments without using LP relaxations. However, they have larger dependencies on k as summarized in Table 1.

The main contribution of this paper is giving an algorithm that computes half-integral farthest solutions to those LPs in time linear in the input size. Because of the connection of computing half-integral LP solutions and FPT algorithms, we automatically obtain linear-time FPT algorithms for various problems, which are summarized in Table 1. In particular, for GROUP FVS and NON-MONOCROMATIC CYCLE TRANSVERSAL, we obtain the first linear-time FPT algorithms. For SUBSET FVS, NODE MULTIWAY CUT, and NODE UNIQUE LABEL COVER, we substantially improve the dependency on the parameter. We note that, for every problem in the table, the $f(k)$ part in the running time of our algorithm matches the known smallest. All of these results are obtained by the same approach, i.e., the branch-and-bound framework combined with the efficient computation of half-integral LPs, which demonstrates the generality of our approach.

1.2 A-path Packing

Solving the half-integral LPs used in the branch-and-bound FPT algorithms is as hard as solving the dual of a natural LP relaxation of a kind of path-packing problem called *non-returning A-path packing* in the following sense. Our LPs are in the following form (called the \mathcal{F} -covering problem): we are given a graph $G = (V, E)$, a root $s \in V$ (which is a vertex obtained by shrinking the vertices whose values are fixed to zero through branching so far), and a set \mathcal{F} of (possibly non-simple) forbidden cycles passing through s , we want to find a function $x : V \rightarrow \mathbb{R}_{\geq 0}$ minimizing the total value $\sum_{v \in V} x(v)$ under the constraints that $x(s) = 0$ and $\sum_{v \in C} x(v) \geq 1$ for every cycle $C \in \mathcal{F}$,

³ Note that edge-deletion problems are easily reducible to the corresponding vertex-deletion problems in most cases by subdividing the edges and by creating k copies of the original vertices.

where we simply denote that a cycle C intersects a vertex v by $v \in C$. The dual of this LP is written down as follows (called the \mathcal{F} -packing problem): we want to find a function $y : \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ maximizing the total value $\sum_{C \in \mathcal{F}} y(C)$ subject to $\sum_{C \in \mathcal{F}: v \in C} y(C) \leq 1$ for every $v \in V \setminus \{s\}$. By setting A as the set of neighbors of s and by imposing the integrality on y , this problem becomes an A -path⁴ packing problem constrained by \mathcal{F} , and it is known that this problem becomes non-returning A -path packing if and only if \mathcal{F} enjoys some nice property, which is satisfied in our setting (see Section 2.2 for details). Moreover, we can observe that, for any instance of non-returning A -path packing, by inserting a new vertex s adjacent to each vertex in A , the dual of its LP relaxation becomes an instance of the LP for NODE UNIQUE LABEL COVER.

As seen in the next subsection, instead of directly computing a minimum \mathcal{F} -cover x , we solve the \mathcal{F} -packing problem, which is equivalent to half-integral non-returning A -path packing. Below, we give a review of existing research on non-returning A -path packing and its special cases.

Finding a maximum fully vertex-disjoint A -path packing and maximum non-bipartite matching are equivalent in the following sense; a matching of a graph forms fully vertex-disjoint V -paths, and moreover, there also exists a simple linear-time reduction from fully vertex-disjoint A -path packing to non-bipartite matching. Mader [25] gave a good characterization for the maximum number of internally disjoint A -paths⁵ in a graph by a min-max formula that commonly generalizes the Tutte–Berge formula for matchings and Menger’s theorem for disjoint paths between two specified vertices (see, e.g., [33]). Polynomial-time algorithms for finding a maximum packing of internally disjoint A -paths were given by Lovász [23, 24] and Schrijver [33, Section 73.1a] via a reduction to (linear) matroid matching. Recently, a further generalized framework of path-packing was introduced (called *packing non-zero/non-returning A -paths*), and its combinatorial properties and algorithms have been investigated [6, 7, 28, 30, 34, 36].

A natural LP relaxation for internally-disjoint A -path packing is as follows: to find a function $y : \mathcal{P} \rightarrow \mathbb{R}_{\geq 0}$ maximizing $\sum_{P \in \mathcal{P}} y(P)$ subject to $\sum_{P \in \mathcal{P}: v \in P} y(P) \leq 1$ for every $v \in V \setminus A$, where \mathcal{P} denotes the set of all A -paths in G . The dual of this LP coincides with a natural LP relaxation for NODE MULTIWAY CUT, and this dual LP is used in the branch-and-bound FPT algorithm for this problem [9]. Due to Garg et al. [14] and Pap [29, 31], it is known that both LPs always enjoy half-integral optimal solutions even if each non-terminal vertex has an individual integral capacity instead of 1. Hirai [16] and Pap [29, 31] developed algorithms for finding such half-integral optimal solutions, which both run in strongly polynomial time (i.e., the number of elementary operations performed through each algorithm does not depend on the capacity values); one makes use of a sophisticated algorithm for the submodular flow problem [13], and the other relies on the ellipsoid method to solve LPs whose coefficient matrices have only $0, \pm 1$ entries [12]. Specialized to the uncapacitated case, Babenko [1] provided a rather simple $O(knm)$ -time algorithm for finding a maximum half-integral packing, where $n = |V|$, $m = |E|$, and k denotes the optimal value, which is at most $O(n)$. Note that, if the capacity values are all finite, then the capacitated case reduces to the uncapacitated case by making as many copies of each non-terminal vertex as its capacity.

From the viewpoint of this field, one of our contributions is giving a combinatorial algorithm for finding a maximum half-integral packing of non-returning A -paths together with a half-integral LP-dual optimal solution (\mathcal{F} -covering). The algorithm requires $O(km)$ calls of a certain oracle, whose computational time is bounded by a constant in many special cases including the above

⁴An A -path is a path between two distinct terminals in a prescribed terminal set A whose inner vertices are all non-terminal vertices (i.e., in $V \setminus A$).

⁵They can share terminals but neither inner vertex nor edge.

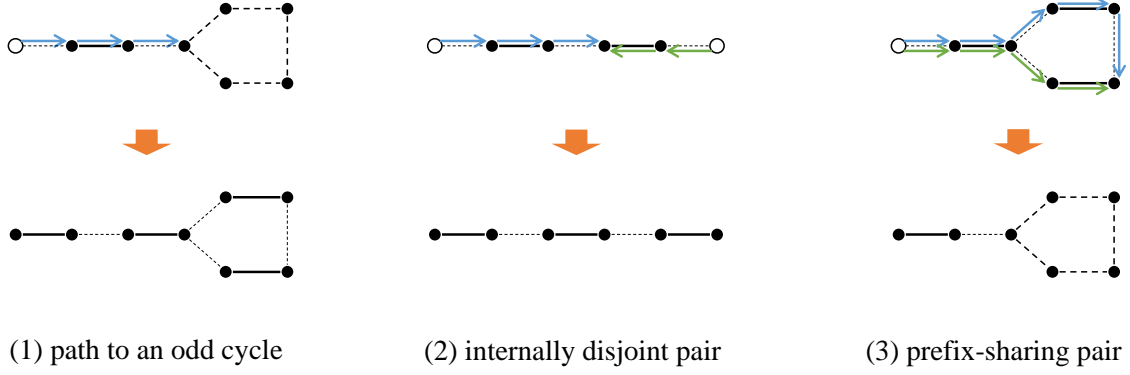


Figure 1: Augmentations for half-integral matching. Thick solid lines denote edges of weight 1, thick dotted lines denote edges of weight $\frac{1}{2}$, and thin dotted lines denote edges of weight 0

internally-disjoint A -paths setting. This essentially improves the computational time even for this special case, and moreover, our algorithm is applicable to much broader situations. We believe that our algorithm for half-integral non-returning A -path packing will lead to a faster algorithm for integral non-returning A -path packing.

1.3 Proof Idea

Instead of directly computing a minimum \mathcal{F} -cover x , we iteratively augment a feasible solution y to the dual problem, i.e., \mathcal{F} -packing. Since the \mathcal{F} -covering and \mathcal{F} -packing problems are the dual LPs to each other, their optimal values are equal. If no augmentation is possible, we can construct a half-integral minimum \mathcal{F} -cover.

We describe the idea behind our algorithm by showing a relation to computing half-integral (non-bipartite) matchings, which are often called *2-matchings* in the field of combinatorial optimization (see, e.g., [33, Chapter 30] for the basics). Recall that matching reduces to A -path packing by setting $A = V$ and A -path packing reduces to \mathcal{F} -packing by inserting a root s adjacent to all the vertices in A . Although we can easily obtain a maximum half-integral matching by a reduction to maximum bipartite matching [26], we use a different approach here. We focus on special half-integral matchings that consist of vertex-disjoint edges of weight 1 and odd cycles⁶ of weight $\frac{1}{2}$. It is known that there always exists a maximum half-integral matching with this special structure [2]. In each step, we search for an alternating path (a path that uses alternately edges of weight 0 and 1; it never uses edges of weight $\frac{1}{2}$) from vertices not used in the current matching by a standard graph search algorithm (e.g., DFS or BFS). There are three types of augmentation (see Figure 1).

The first case is when we have found an alternating path P of length $2a + 1$ ending at a vertex on an odd cycle C of length $2b + 1$. The current matching uses even edges in P with weight 1 and all the edges in C with weight $\frac{1}{2}$. Thus, the size of the current matching induced by P and C is $a + b + \frac{1}{2}$. We can easily augment this matching to an integral matching of size $a + b + 1$ by alternately taking the edges in P and C .

The second case is when we have found a pair of internally disjoint alternating paths P of odd length and Q of even length ending at the same vertex but starting from distinct vertices. In this case, the concatenation of P and Q becomes an alternating path connecting two vertices not used

⁶The number of edges in the cycle is odd.

in the current matching. Thus, by the standard augmentation, we can augment the matching size by 1.

The third case is when we have found a pair of prefix-sharing alternating paths P of odd length and Q of even length ending at the same vertex. This case corresponds to a *blossom* in Edmonds' algorithm for the maximum (integral) matching problem [11]. In Edmonds' algorithm, the alternation is applied to the common prefix and then the vertices on the cycles induced by P and Q are contracted. In our approach, by applying the alternation to the common prefix, and then by transforming the cycle induced by P and Q to an odd cycle of weight $\frac{1}{2}$, we can augment the matching size by $\frac{1}{2}$.

In our algorithm for \mathcal{F} -packing/covering, we use a similar approach. We focus on a special type of half-integral \mathcal{F} -packings that consist of internally disjoint cycles in \mathcal{F} of weight 1 (called *integral cycles*) and *wheels*, which are sums of an odd number of cycles in \mathcal{F} of weight $\frac{1}{2}$ and correspond to odd cycles for matching. See Section 3.1.1 for the precise definition. Although it is known that there always exists a maximum half-integral packing with a similar special structure for internally disjoint A -paths [27], the existence of such special solution has been previously not known for non-returning A -paths. The correctness of our algorithm gives a constructive proof of the existence.

In each step, we search for an alternating path from s . As opposed to the case of computing half-integral matchings, alternating paths may use edges contained in wheels. In the following explanation, however, we ignore such a case for simplicity. Roughly speaking, an alternating path is a sequence of paths P_1, \dots, P_l , where P_i is a path internally disjoint from any integral cycle and wheel for odd i and is fully contained in an integral cycle for even i . In the alternation operation, we replace the integral cycle containing P_2 with the cycle obtained by concatenating P_1 and the path from s to the first vertex of P_2 along the integral cycle in the same direction as P_2 , we replace the integral cycle containing P_4 with \dots , and so on. For ensuring that each introduced cycle is in \mathcal{F} , the definition of alternating paths is rather complicated; however, it essentially plays the same role as that for the half-integral matching case.

There are three types of augmentation, each of which corresponds to the one for the half-integral matching case. When we find an alternating path ending at a vertex on a wheel which is a sum of $2a + 1$ cycles in \mathcal{F} of weight $\frac{1}{2}$, we can augment the \mathcal{F} -packing by $\frac{1}{2}$ by decomposing the wheel into a integral cycles and by introducing a new integral cycle. We call such an alternating path as an *augmenting path*. When we find a pair of internally disjoint alternating paths P and Q ending at the same vertex (with some additional conditions), we can augment the \mathcal{F} -packing by 1 by applying the alternation to each of P and Q and then by introducing a new integral cycle. When we find a pair of prefix-sharing alternating paths P and Q ending at the same vertex (with some additional conditions), we can augment the \mathcal{F} -packing by $\frac{1}{2}$ by applying the alternation to the common prefix and then by introducing a new wheel. We call these two types of a pair of alternating paths as an *augmenting pair*.

As the definition of alternating paths changes, we cannot use standard graph search algorithms for finding augmenting paths/pairs. Instead, we give a new search algorithm that runs in $O(m)$ time and requires only $O(m)$ membership tests for \mathcal{F} . To achieve the linear time complexity, we need to test membership in constant time. By exploiting an observation that the algorithm only tests membership against some special cycles, we achieve constant-time membership tests for the problems we consider.

1.4 Comparison to Babenko's Algorithm

Because our algorithm improves the previous best running time even against the special case, i.e., internally-disjoint A -path packing, by Babenko [1], we compare our algorithm with Babenko's algorithm to clarify the reason that we obtain such an improvement. The main difference is the existence of augmenting pairs and the algorithm for computing augmenting paths/pairs.

In our algorithm, we focus on packings with a special structure. Both the definition of alternating paths and the algorithm for searching augmenting paths/pairs strongly rely on this structure. While the existence of a maximum half-integral packing with the special structure was already known for internally-disjoint A -path packing [27], Babenko's algorithm does not directly exploit the structure but uses a much weaker structure. This is because his augmentation strategy does not preserve the special structure because it does not consider a notion corresponding to augmenting pairs of our algorithm.

In our algorithm, we directly compute an augmenting path/pair in linear time. On the other hand, in Babenko's algorithm, an auxiliary network and its s - t flow f are constructed from the current packing and then an f -augmenting path is computed by the standard DFS algorithm. From the obtained f -augmenting path, either we can augment the current packing or we can find a set of vertices that can be safely contracted to some vertex in A . Because contractions occur at most $O(n)$ time per augmentation, the running time becomes $O(knm)$.

1.5 Organization

We introduce notions used throughout the paper in Section 2. In Section 3, we show a fast algorithm that computes an optimal half-integral solution to the \mathcal{F} -packing problem and transforms it into an optimal half-integral solution to the \mathcal{F} -covering problem. Using this algorithm, we give linear-time FPT algorithms in Section 4.

2 Definitions

2.1 Basic Notations

For a multiset S on the ground set U , the *multiplicity function* $\mathbf{1}_S : U \rightarrow \mathbb{Z}_+$ is defined so that $\mathbf{1}_S(a)$ is the number of times that $a \in U$ appears in S . For two multisets A and B on the same ground set U , we denote by $A \setminus B$ the multiset such that $\mathbf{1}_{A \setminus B}(a) = \max\{\mathbf{1}_A(a) - \mathbf{1}_B(a), 0\}$ holds for any element $a \in U$. For a function $f : U \rightarrow \mathbb{R}$ and a multiset S on the ground set U , we define $f(S) := \sum_{a \in U} \mathbf{1}_S(a)f(a)$. For a condition C , we define $[C] \in \{0, 1\}$ as $[C] = 1$ if and only if the condition C holds.

All the graphs in this paper are undirected; however, we sometimes need to take care of the direction of edges. For an undirected graph $G = (V, E)$, we use the symbol \hat{E} when we take care of the direction of the edges, i.e., $uv = vu$ for $uv \in E$ but $uv \neq vu$ for $uv \in \hat{E}$. For simplicity, we assume the graphs are simple; if a graph contains multiple edges or self-loops, we can easily obtain an equivalent simple graph by subdividing the edges. For vertex $v \in V$, we denote the set of incident edges by $\delta(v)$.

For an undirected graph $G = (V, E)$, we define a *walk* in G as an ordered list $W = (v_0, \dots, v_l)$ of vertices such that $v_{i-1}v_i \in E$ for all $i = 1, \dots, l$. The integer l is called the *length* of the walk. We denote the first and last vertices of W by $s(W) = v_0$ and by $t(W) = v_l$, respectively, and we say

that W starts from $s(W)$ and ends at $t(W)$. We denote the multisets of vertices, of inner vertices, and of (undirected) edges appeared in W by $V(W) = \{v_0, \dots, v_l\}$, by $V_{\text{in}}(W) = \{v_1, \dots, v_{l-1}\} = V(W) \setminus \{s(W), t(W)\}$, and by $E(W) = \{v_0v_1, \dots, v_{l-1}v_l\}$, respectively. For an edge $e = uv \in \hat{E}$, we simply use the same symbol e to denote the walk (u, v) . A walk W is called a *path* if $\mathbf{1}_{V(W)}(v) \leq 1$ for every $v \in V$. A walk W is called a (*simple*) *cycle* if $l \geq 3$, $s(W) = t(W)$, and $\mathbf{1}_{V(W) \setminus \{s(W)\}}(v) \leq 1$ for every $v \in V$ (which implies $\mathbf{1}_{V(W)}(s(W)) = 2$). For two walks W and W' , we say that W is *internally disjoint from* W' if they share no edges (i.e., $\mathbf{1}_{E(W)}(e) \cdot \mathbf{1}_{E(W')}(e) = 0$ for every $e \in E$) and no inner vertices of W appear in W' (i.e., $\mathbf{1}_{V_{\text{in}}(W)}(v) \cdot \mathbf{1}_{V(W')}(v) = 0$ for every $v \in V$)⁷. For a walk $W = (v_0, \dots, v_l)$, we define the *reversed walk* as $W^{-1} = (v_l, \dots, v_0)$. For a walk $W_1 = (v_0, \dots, v_{l'})$ and a walk $W_2 = (v_{l'}, \dots, v_l)$ (where $0 \leq l' \leq l$), we define the *concatenation* of the two walks as $W_1 \circ W_2 = (v_0, \dots, v_l)$. The notation $W_1 \circ W_2$ implicitly implies $t(W_1) = s(W_2)$.

2.2 \mathcal{F} -Packing/Covering

Let $G = (V, E)$ be a graph and fix a root $s \in V$. A walk starting from s is called an *s-walk* and an *s-walk* $W = (v_0, \dots, v_l)$ is called an *s-path* if (v_0, \dots, v_{l-1}) is a path and $v_l \neq v_{l-2}$.⁸ A positive-length *s-walk* ending at s is called an *s-cycle*. Note that an *s-path* and an *s-cycle* may not be simple (some inner vertices may be visited more than once). Let \mathcal{F} be a (possibly infinite) set of *s-cycles*. We may assume that \mathcal{F} is given as a *membership oracle*: if we specify an *s-cycle* C , then it answers whether C belongs to \mathcal{F} or not. A function $x : V \rightarrow \mathbb{R}_{\geq 0}$ is called an \mathcal{F} -*cover* if $x(s) = 0$ and $x(V(C)) \geq 1$ for every $C \in \mathcal{F}$. The size of an \mathcal{F} -cover x is defined as $|x| = x(V)$. A function $y : \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ is called an \mathcal{F} -*packing* if for every vertex $v \in V \setminus \{s\}$, it holds that $\sum_{C \in \mathcal{F}} \mathbf{1}_{V(C)}(v) y(C) \leq 1$. The size of an \mathcal{F} -packing y is defined as $|y| = y(\mathcal{F})$. The minimum size of an \mathcal{F} -cover and the maximum size of an \mathcal{F} -packing coincide because the corresponding LP problems are dual to each other.

For two *s-walks* P and Q ending at the same vertex, we write $P \equiv Q$ if and only if $P \circ Q^{-1} \notin \mathcal{F}$. \mathcal{F} is called *nice* if the relation (\equiv) is an equivalence relation, i.e., (i) $P \equiv P$, (ii) $P \equiv Q \iff Q \equiv P$, and (iii) $P \equiv Q \wedge Q \equiv R \implies P \equiv R$ hold for every *s-walks* P , Q , and R ending at the same vertex. We note that for a nice set \mathcal{F} , \mathcal{F} -packing is equivalent to non-returning *A-path* packing (cf. [27, pp. 109–111]). In this paper, we use this problem formulation instead of *A-path* packing for convenience. We will often use the following properties of a nice set \mathcal{F} .

Lemma 1. *Let A, B be *s-cycles*, P, Q be *s-walks*, W be a walk, and C be a cycle. For any nice set \mathcal{F} , the following holds.*

1. $A \in \mathcal{F} \iff A^{-1} \in \mathcal{F}$.
2. If $A \equiv B$, then $A \in \mathcal{F} \iff B \in \mathcal{F}$.
3. If $A \not\equiv B$, then at least one of A and B is in \mathcal{F} .
4. $P \circ W \equiv Q \iff P \equiv Q \circ W^{-1}$.
5. If $P \equiv Q$, then $P \circ W \equiv Q \circ W$.
6. If $P \equiv Q$, then $P \circ C \circ P^{-1} \in \mathcal{F} \iff Q \circ C \circ Q^{-1} \in \mathcal{F}$.

⁷For convenience, we adopt this asymmetric definition.

⁸For convenience, we allow that the last vertex of *s-path* W can be identical to another vertex on W .

Proof. The claims can be proved as follows.

1. $A \in \mathcal{F} \iff A \not\equiv (s) \iff (s) \not\equiv A \iff A^{-1} \in \mathcal{F}.$
2. $A \in \mathcal{F} \iff A \not\equiv (s) \iff B \not\equiv (s) \iff B \in \mathcal{F}.$
3. $A \notin \mathcal{F} \implies B \not\equiv A \equiv (s) \implies B \in \mathcal{F}.$
4. $P \circ W \equiv Q \iff P \circ W \circ Q^{-1} \notin \mathcal{F} \iff P \equiv Q \circ W^{-1}.$
5. $P \circ W \equiv P \circ W \implies P \circ W \circ W^{-1} \equiv P \implies P \circ W \circ W^{-1} \equiv Q \implies P \circ W \equiv Q \circ W.$
6. $P \circ C \circ P^{-1} \in \mathcal{F} \iff Q \circ C \circ P^{-1} \in \mathcal{F} \iff P \circ C^{-1} \circ Q^{-1} \in \mathcal{F} \iff Q \circ C^{-1} \circ Q^{-1} \in \mathcal{F} \iff Q \circ C \circ Q^{-1} \in \mathcal{F}.$ \square

We note that, from the first property, we can ignore the direction of s -cycles when testing membership in \mathcal{F} .

2.3 Single-Branching Pair and Equivalence Oracle

A pair (P, Q) of s -paths ending at the same vertex is called *single-branching* if they can be written as $P = R \circ P'$ and $Q = R \circ Q'$ for a path R and two walks P' and Q' for which $P' \circ Q'^{-1}$ forms a simple cycle that is internally disjoint from R . Here, each of R , P' , and Q' can be a path of length zero, but the cycle $P' \circ Q'^{-1}$ must have a positive length. An *equivalence oracle* for \mathcal{F} is an algorithm that, given a single-branching pair (P, Q) , answers whether $P \equiv Q$ or not.

In our algorithm, we use the following implementation of the equivalence oracle. Let $U \ni \epsilon$ be a set with a special element ϵ . We design a pair of functions *append* $\mathcal{A} : U \times \hat{E} \rightarrow U$ and *test* $\mathcal{T} : U \times U \rightarrow \{0, 1\}$ such that, for any single-branching pair (P, Q) , $\mathcal{T}(\mathcal{A}^*(P), \mathcal{A}^*(Q)) = 1$ if and only if $P \equiv Q$, where $\mathcal{A}^*(W)$ is defined as $\mathcal{A}^*((s)) := \epsilon$ and $\mathcal{A}^*(W \circ e) := \mathcal{A}(\mathcal{A}^*(W), e)$. In general, these functions can be implemented by using a membership oracle \mathcal{M} as follows. Let U be the set of all s -paths and $\epsilon = (s)$. The append function $\mathcal{A}(W, e)$ returns the concatenation $W \circ e$, and the test function $\mathcal{T}(P, Q)$ returns $[P \circ Q^{-1} \notin \mathcal{F}]$. The append function takes $O(n)$ time and the test function takes $O(n + T)$ time, where T is the time for the membership oracle. For special cases, we can implement the functions \mathcal{A} and \mathcal{T} more efficiently. We give only one example here. Other examples are deferred to Section 4.1.

The LP for SUBSET FEEDBACK VERTEX SET is the \mathcal{F} -covering problem such that, for a single-branching pair (P, Q) , $P \equiv Q$ if and only if the simple cycle contained in $P \circ Q^{-1}$ does not contain any vertex in the input subset S . Let $U = E \cup \{\epsilon\}$, which will represent the last passed edge incident to S . The append function is defined as $\mathcal{A}(a, e) = e$ if e is incident to S and otherwise $\mathcal{A}(a, e) = a$. The test function is defined as $\mathcal{T}(a, b) = [a = b]$. Both the functions take a constant time.

3 Half-Integral Packing and Covering

In this section, we prove the following theorem.

Theorem 1. *Given a graph of m edges and an equivalence oracle of a nice set \mathcal{F} implemented by $T_{\mathcal{E}}$ -time append/test functions, a pair of a maximum \mathcal{F} -packing and a minimum \mathcal{F} -cover of size k can be computed in $O(kmT_{\mathcal{E}})$ time.*

Corollary 1. *Given a graph of n vertices and m edges and a $T_{\mathcal{M}}$ -time membership oracle of a nice set \mathcal{F} , a pair of a maximum \mathcal{F} -packing and a minimum \mathcal{F} -cover of size k can be computed in $O(km(n + T_{\mathcal{M}}))$ time.*

Our algorithm is based on a simple augmentation strategy summarized as follows. Starting with $y(C) = 0$ for every $C \in \mathcal{F}$, we repeatedly update a half-integral \mathcal{F} -packing y that always consists of only two types of s -cycles defined in Section 3.1.1. In each iteration, we search an *augmenting path/pair* (see Section 3.1.2) by Algorithm 1 described in Section 3.2. If one is found, we can improve the current \mathcal{F} -packing y in linear time by Lemmas 2 and 3 (Augmentation); otherwise, as shown in Section 3.3, we can naturally construct a half-integral \mathcal{F} -cover of size $|y|$, which guarantees the optimality of y with the aid of the LP-duality. Since each augmentation increases $|y|$ by at least $\frac{1}{2}$, the number of iterations is bounded by $O(k)$, where k is the optimal value. Algorithm 1 can be implemented in linear time by Lemma 6, which concludes Theorem 1.

3.1 Preliminaries

3.1.1 Basic \mathcal{F} -Packing

In what follows, we focus on \mathcal{F} -packings that consist of only two types of s -cycles. One is a simple cycle $C \in \mathcal{F}$ of weight 1, which is called an *integral cycle*. The other is a *wheel* defined as follows, which consists of an odd number of s -cycles $C \in \mathcal{F}$ of weight $\frac{1}{2}$.

Definition 1. A pair of a simple cycle $C = H_1 \circ \dots \circ H_d$ of weight $\frac{1}{2}$ and paths $\{S_1, \dots, S_d\}$ of weight 1 is called a *wheel* if it satisfies the following conditions (see Figure 2).

1. d is an odd positive integer.
2. $s \notin V(C)$.
3. For any i , S_i is a path of positive length from s to $s(H_i) = t(H_{i-1})$ (where $H_0 = H_d$) that is internally disjoint from C .
4. For any distinct i and j , S_i and S_j intersect only at the root s .
5. For any i , $S_i \circ H_i \circ S_{i+1}^{-1} \in \mathcal{F}$, where $S_{d+1} = S_1$.

The integer d is called the *degree* of the wheel, the cycle C is called the *half-integral cycle* of the wheel, and the paths $\{S_1, \dots, S_d\}$ are called the *spokes* of the wheel.

Note that a wheel of degree 1 is an s -cycle $S_1 \circ C \circ S_1^{-1} \in \mathcal{F}$ of weight $\frac{1}{2}$ such that the path S_1 has a positive length, and a wheel of degree $d \geq 3$ is a sum of d simple cycles $\{S_1 \circ H_1 \circ S_2^{-1}, \dots, S_d \circ H_d \circ S_1^{-1}\} \subseteq \mathcal{F}$ of weight $\frac{1}{2}$. Even when $d \geq 3$, a wheel can be regarded as one s -cycle obtained by concatenating those d simple cycles, which may not be in \mathcal{F} .

An \mathcal{F} -packing y is called a *basic \mathcal{F} -packing* if it is a sum of integral cycles and wheels⁹ whose half-integral cycles are completely disjoint in the sense that they share neither vertex nor edge. In other words, for a basic \mathcal{F} -packing y , each vertex other than the root s is contained in at most one of the integral cycles and the wheels, and each edge is in at most one of the integral cycles,

⁹Since y is an \mathcal{F} -packing, any integral cycle and any spoke in y can intersect with another integral cycle or wheel in y only at the root s .

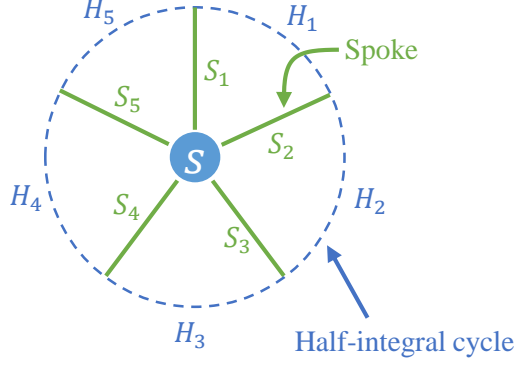


Figure 2: Wheel of degree 5.

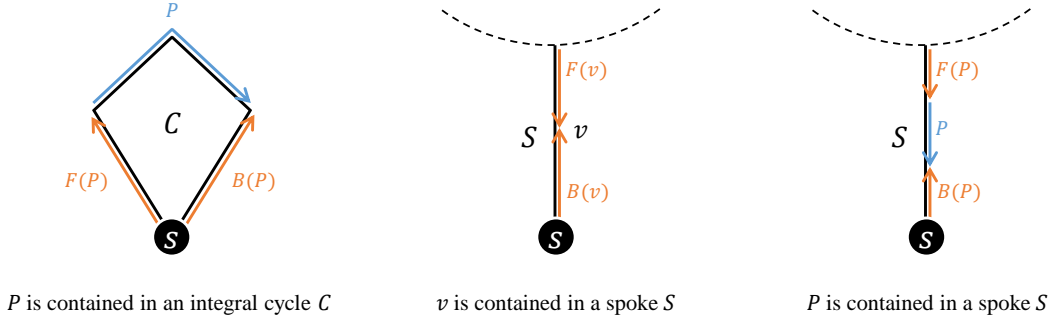


Figure 3: Definition of F_y and B_y .

the spokes, and the half-integral cycles. In our algorithm, a basic \mathcal{F} -packing y is dealt with as a weighted graph¹⁰ so that we can efficiently update integral cycles and wheels in y . We denote by $V(y)$ and $E(y)$ the sets of vertices and (undirected) edges, respectively, contained in some integral cycle or wheel in y (i.e., of positive weights), and in particular by $V_1(y)$ and $E_1(y)$ the sets of those which are contained in some integral cycle or spoke in y (i.e., are of weight at least 1).

For a basic \mathcal{F} -packing y , we define two functions F_y (Forward) and B_y (Backward) as follows (see Figure 3). For a path P contained in an integral cycle C in y , we denote by $F_y(P)$ the path from s to $s(P)$ along C in the same direction as P and by $B_y(P)$ the path from s to $t(P)$ along C in the opposite direction to P (i.e., $F_y(P) \circ P \circ B_y(P)^{-1} = C$). For a vertex v contained in a spoke S in y , we denote by $F_y(v)$ be the path from $t(S)$ to v along S and by $B_y(v)$ the path from s to v along S (i.e., $F_y(v) \circ B_y(v)^{-1} = S^{-1}$). For a path P contained in a spoke S in y in the opposite direction to S , we define $F_y(P) := F_y(s(P))$ and $B_y(P) := B_y(t(P))$ (i.e., $F_y(P) \circ P \circ B_y(P)^{-1} = S^{-1}$). We omit the subscript y if it is clear from the context.

3.1.2 Augmenting Path/Pair

To define our *augmenting path/pair*, we first define an *alternating path* as follows.

¹⁰ The weight is naturally defined for each vertex v and edge e by $\sum_{C \in \mathcal{F}} \mathbf{1}_{V(C)}(v)y(C)$ and $\sum_{C \in \mathcal{F}} \mathbf{1}_{E(C)}(e)y(C)$, respectively.

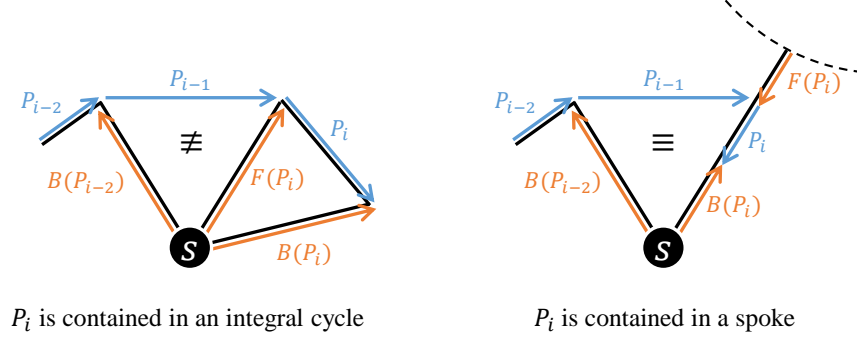


Figure 4: Conditions for alternating paths.

Definition 2. For a basic \mathcal{F} -packing y , a concatenation of paths $P = P_1 \circ \dots \circ P_l$ is called a y -alternating path if it satisfies all the following conditions.

1. Edges in $E(P)$ are distinct, i.e., $\mathbf{1}_{E(P)}(e) \leq 1$ for every $e \in E$.
2. Every vertex in P that is not contained in any integral cycle or spoke in y appears in P at most once¹¹, i.e., $\mathbf{1}_{V(P)}(v) \leq 1$ for every $v \in V \setminus V_1(y)$.
3. Each P_i is a path of positive length satisfying the following conditions.
 - (a) For any odd i , P_i is internally disjoint from every integral cycle and wheel in y .
 - (b) For any even i , P_i is contained in an integral cycle or a spoke in y . In the latter case, the direction of P_i is toward the root s .
4. Let us define $B(P_0) := (s)$. The following conditions are satisfied for any even $i \geq 2$ ¹² (see Figure 4).
 - (a) If P_i is contained in an integral cycle, $B(P_{i-2}) \circ P_{i-1} \neq F(P_i)$ and none of the P_j 's are contained in $B(P_i)$ for $j > i$. Moreover, if $B(P_{i-2}) \circ P_{i-1} \neq B(P_i) \circ P_i^{-1}$, none of the P_j 's are contained in $F(P_i)$ for $j > i$.
 - (b) If P_i is contained in a spoke, $B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}$ and none of the P_j 's are contained in $B(P_i)$ for $j > i$.

Each P_i is called a *segment* of P . If P consists of only a single segment, it is called *single-segment*.

For a y -alternating path $P = P_1 \circ \dots \circ P_l$, we define $T_y(P)$ (Tail) as $T_y(P) := B_y(P_{l-1}) \circ P_l$ if l is odd and $T_y(P) := B_y(P_l)$ if l is even. We omit the subscript y if it is clear from the context.

Definition 3. A y -alternating path $P = P_1 \circ \dots \circ P_l$ is called a y -augmenting path if l is odd and one of the following conditions is satisfied.

¹¹ From the other properties of y -alternating paths, a vertex contained in an integral cycle or a spoke can appear twice in P . Hence P may not be a path in the precise sense.

¹² Technically, the conditions “none of the P_j 's are contained in ...” say that P does not admit a *shortcut*; e.g., if there is some P_j with $j > i$ contained in $B(P_i)$, we can obtain another y -alternating path $P_1 \circ \dots \circ P_{i-1} \circ W \circ P_{j+1} \circ \dots \circ P_l$, where W is the path from $s(P_i)$ to $t(P_j)$ along the integral cycle or spoke. As in the case of matroid intersection, we need a shortcut-less alternating path.

1. $t(P)$ is contained in a half-integral cycle but in no spokes, i.e., $t(P) \in V(y) \setminus V_1(y)$.
2. $t(P)$ is contained in a spoke S_i and the following two conditions are satisfied.
 - (a) $T(P) \neq B(t(P))$.
 - (b) For any P_j contained in the spoke S_i , $t(P)$ is contained in $F(P_j)$.

Definition 4. Let $P = P_1 \circ \dots \circ P_p$ and $Q = Q_1 \circ \dots \circ Q_q$ be a pair of y -alternating paths ending at the same vertex. (P, Q) is called a *y-augmenting pair* if it satisfies all the following conditions.

1. P and Q can be written as $P = R \circ P'$ and $Q = R \circ Q'$ for some walk R such that P' and Q' share no edges.
2. $T(P) \neq T(Q)$.
3. At least one of p and q is odd, and if both of p and q are odd, $t(P) \notin V(y)$.
4. For any P_i contained in an integral cycle, none of the Q_j 's are contained in $B(P_i)$ in the opposite direction. Moreover, if $B(P_{i-2}) \circ P_{i-1} \neq B(P_i) \circ P_i^{-1}$, none of the Q_j 's are contained in $F(P_i)$ in the same direction. The symmetric condition holds for any Q_i contained in an integral cycle.

Note that for any y -alternating path P , the condition 4 is always satisfied against (P, P) . Therefore, for testing the condition 4 against (P, Q) , we only need to test pairs (P_i, Q_j) such that at least one of P_i or Q_j is not contained in the common prefix R .

Using a y -augmenting path/pair, we can improve a basic \mathcal{F} -packing y in linear time by the following lemmas, whose proofs are given in Section 3.4.

Lemma 2. *Given a basic \mathcal{F} -packing y and a y -augmenting path, a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$ can be constructed in linear time.*

Lemma 3. *Given a basic \mathcal{F} -packing y and a y -augmenting pair, a basic \mathcal{F} -packing of size at least $|y| + \frac{1}{2}$ can be constructed in linear time.*

3.2 Finding Augmenting Path/Pair

In this subsection, we propose an algorithm for computing a y -augmenting path or pair. Algorithm 1 describes a rough sketch of the algorithm; we describe the detail of an efficient implementation later. Note that in this subsection, we prove only the soundness of the algorithm (i.e., the algorithm never returns a path or pair which is not y -augmenting). The completeness of the algorithm (i.e., the algorithm always finds a y -augmenting path or pair if exists) follows from Lemma 7 proved in the next subsection.

In the algorithm, we hold two indices $a(C)$ and $b(C)$ for each integral cycle C initialized as $a(C) = 1$ and $b(C) = l - 1$, where l is the length of C , an index $a(S)$ for each spoke S initialized as $a(S) = 1$, a set of active vertices X initialized as $X = \{s\}$, and an s -walk $P(v)$ for each *visited* vertex, which represents the path along the search tree; here, we call a vertex v visited if v has been ever pushed into X . We define *boundaries* as the set of vertices consisting of $v_{a(C)}$ and $v_{b(C)}$ for each integral cycle $C = (v_0, \dots, v_l)$ and $v_{a(S)}$ for each spoke $S = (v_0, \dots, v_l)$. During the execution of the algorithm, we preserve the following invariants.

Algorithm 1 Algorithm for computing a y -augmenting path/pair

```
1: Initialize  $a(C) \leftarrow 1$  and  $b(C) \leftarrow l - 1$  for each integral cycle  $C$  of length  $l$ .
2: Initialize  $a(S) \leftarrow 1$  for each spoke  $S$ .
3: Initialize  $P(s) \leftarrow (s)$  and  $X \leftarrow \{s\}$ .
4: while  $X \neq \emptyset$  do
5:   Pick a vertex  $u \in X$  and remove  $u$  from  $X$ .
6:   for  $e = uv \in \delta(u) \setminus E(y)$  do
7:     if  $v$  is visited then
8:       if  $T(P(v)) \neq T(P(u)) \circ e$  then return  $(P(v), P(u) \circ e)$ 
9:     else
10:      if  $v \notin V(y)$  then
11:         $P(v) \leftarrow P(u) \circ e$  and  $X \leftarrow X \cup \{v\}$ .
12:      else if  $v$  is contained in an integral cycle  $C = (v_0, \dots, v_l)$  then
13:        Let  $i$  be the index such that  $v_i = v$ .
14:        if  $T(P(u) \circ e) \neq (v_l, \dots, v_i)$  then
15:          for  $j \in \{a(C), \dots, i - 1\}$  do
16:             $P(v_j) \leftarrow (P(u) \circ e) \circ (v_i, \dots, v_j)$  and  $X \leftarrow X \cup \{v_j\}$ .
17:           $a(C) \leftarrow i$ .
18:        if  $T(P(u) \circ e) \neq (v_0, \dots, v_i)$  then
19:          for  $j \in \{i + 1, \dots, b(C)\}$  do
20:             $P(v_j) \leftarrow (P(u) \circ e) \circ (v_i, \dots, v_j)$  and  $X \leftarrow X \cup \{v_j\}$ .
21:           $b(C) \leftarrow i$ .
22:        else if  $v$  is contained in a spoke  $S = (v_0, \dots, v_l)$  then
23:          Let  $i$  be the index such that  $v_i = v$ .
24:          if  $T(P(u) \circ e) \neq (v_0, \dots, v_i)$  then return  $P(u) \circ e$ 
25:          for  $j \in \{a(S), \dots, i - 1\}$  do
26:             $P(v_j) \leftarrow (P(u) \circ e) \circ (v_i, \dots, v_j)$  and  $X \leftarrow X \cup \{v_j\}$ .
27:           $a(S) \leftarrow i$ .
28:        else ( $v$  is contained in a half-integral cycle)
29:          return  $P(u) \circ e$ 
30: return NO
```

Lemma 4. *The following invariants hold at any iteration of Algorithm 1.*

1. *For any visited v , the following holds:*

- (a) $P(v)$ is a y -alternating path,
- (b) $V(P(v))$ contains only visited vertices or boundaries, and
- (c) $P(v)$ has an odd number of segments iff $v \notin V(y)$.

2. *For any visited u and v , $(P(u), P(v))$ satisfies the condition 1 of y -augmenting pairs (Definition 4).*

3. *For any integral cycle $C = (v_0, \dots, v_l)$, the following holds:*

- (a) $a(C) \leq b(C)$,
- (b) v_i is visited iff $i < a(C)$ or $b(C) < i$, and
- (c) for any $P(v)$ with segments $P_1 \circ \dots \circ P_p$ and any segment P_i contained in C , P_i is contained in $(v_{a(C)}, \dots, v_0)$ or $(v_{b(C)}, \dots, v_l)$ in these directions; moreover if $B(P_{i-2}) \circ P_{i-1} \neq B(P_i) \circ P_i^{-1}$ holds, then $s(P_i) = v_{a(C)} = v_{b(C)}$ holds.

4. For any spoke $S = (v_0, \dots, v_l)$, the following holds:

- (a) v_i is visited iff $i < a(S)$ and
- (b) for any $P(v)$, all the segments of $P(v)$ contained in S are contained in $(v_{a(S)}, \dots, v_0)$.

Proof. In each iteration, we pick an arbitrary vertex u from X (line 5). From the invariant 1, $P(u)$ is a y -alternating path. Then, we iterate over the edges $e = uv \in \delta(u) \setminus E(y)$. Note that if e is not contained in $E(P(u))$, $P(u) \circ e$ is also a y -alternating path and $T(P(u) \circ e) = T(P(u)) \circ e$ holds.

When v is already visited and $T(P(v)) \neq T(P(u)) \circ e$ holds, we return a pair $(P(v), P(u) \circ e)$ (line 8).

Claim 1. $(P(v), P(u) \circ e)$ returned at line 8 is a y -augmenting pair.

Proof. First, we prove $e \notin E(P(u))$, which implies that $P(u) \circ e$ is a y -alternating path. If $e \in E(P(u))$, from the condition 2 of y -alternating paths (Definition 2), e^{-1} is either the last edge of $P(u)$ or the last edge of an odd segment P_i of $P(u) := P_1 \circ \dots \circ P_p$. In the former case, $T(P(u)) \circ e = T(P(v)) \circ e^{-1} \circ e \equiv T(P(v))$ holds, which is a contradiction. In the latter case, from the invariant 2 against $(P(u), P(v))$, we have $P(v) = P_1 \circ \dots \circ P_{i-1} \circ W$, where W is the path satisfying $W \circ e^{-1} = P_i$. Because P_{i+1} and P_p are contained in the same integral cycle or spoke in the same direction, we have $B(P_{i-1}) \circ P_i \equiv B(P_{i+1}) \circ P_{i+1}^{-1}$ from the condition 4 of y -alternating paths. Therefore, we have $T(P(u)) \circ e = B(P_p) \circ e = B(P_{i+1}) \circ P_{i+1}^{-1} \circ e \equiv B(P_{i-1}) \circ P_i \circ e = B(P_{i-1}) \circ W \circ e^{-1} \circ e = T(P(v)) \circ e^{-1} \circ e \equiv T(P(v))$, which is a contradiction.

Next, we prove that $(P(v), P(u) \circ e)$ satisfies all the conditions of y -augmenting pairs (Definition 4). From the invariant 2, the condition 1 is satisfied. The condition 2 is satisfied because $T(P(v)) \neq T(P(u)) \circ e = T(P(u) \circ e)$. The condition 3 is satisfied because the number of segments of $P(u) \circ e$ is odd and because, from the invariant 1, $v \in V(y)$ implies that the number of segments of $P(v)$ is even. The condition 4 follows from the invariant 3. \square

When v is not visited, we consider four cases: (Case 1) $v \notin V(y)$, (Case 2) v is contained in an integral cycle, (Case 3) v is contained in a spoke, or (Case 4) v is contained in a half-integral cycle. Note that when v is not visited, e is not contained in $E(P(u))$; therefore, $P(u) \circ e$ is a y -alternating path. In the first case, we set $P(v) \leftarrow P(u) \circ e$ and insert v into X (line 11). Because $e \notin E(y)$ holds, all the invariants are clearly preserved. In the last case, we return a y -augmenting path $P(u) \circ e$ (line 29).

(Case 2) Let $C = (v_0, \dots, v_l)$ be the integral cycle containing v and let i be the index such that $v_i = v$. Because v is not visited, $a(C) \leq i \leq b(C)$ holds from the invariant 3. If $T(P(u) \circ e) \neq (v_l, \dots, v_i)$ holds, we set $P(v_j) \leftarrow (P(u) \circ e) \circ (v_i, \dots, v_j)$ and insert v_j into X for each index $j \in \{a(C), \dots, i-1\}$, and then update $a(C) \leftarrow i$ (lines 14–17). Similarly, if $T(P(u) \circ e) \neq (v_0, \dots, v_i)$ holds, we set $P(v_j) \leftarrow (P(u) \circ e) \circ (v_i, \dots, v_j)$ and insert v_j into X for each index $j \in \{i+1, \dots, b(C)\}$, and then update $b(C) \leftarrow i$ (lines 18–21).

Claim 2. The lines 14–21 preserve all the invariants.

Proof. The invariants 1b, 1c, 2, 3a, 3b, and 4 are clearly preserved. Let $a(C)$ and $b(C)$ denote the indices before the updates. For an index $j \in \{a(C), \dots, i-1\}$, let $P_1 \circ \dots \circ P_p$ be the segments of $P(v_j)$. If $B(P_{p-2}) \circ P_{p-1} \neq B(P_p) \circ P_p^{-1}$ holds, we have $T(P(u) \circ e) = B(P_{p-2}) \circ P_{p-1} \neq B(P_p) \circ P_p^{-1} = (v_0, \dots, v_i)$. Therefore, both of $a(C)$ and $b(C)$ are updated to i . The same argument applies to the case of $j \in \{i+1, \dots, b(C)\}$. Thus, the invariant 3c is preserved.

Finally, we prove the invariant 1a by showing that $P(v_j)$ is a y -alternating path for any newly visited v_j . The conditions 1–3 of y -alternating paths (Definition 2) are clearly satisfied. Consider the case of $j \in \{a(C), \dots, i-1\}$; the proof for the case of $j \in \{i+1, \dots, b(C)\}$ is symmetric. Let $P_1 \circ \dots \circ P_p$ be the segments of $P(v_j)$. Because $B(P_{p-2}) \circ P_{p-1} = T(P(u) \circ e) \neq (v_l, \dots, v_i) = F(P_p)$ holds, the condition 4 is satisfied for p . Let P_k be a segment with $k < p$ contained in the same integral cycle C . From the invariant 3c and because $a(C) \leq j < i \leq b(C)$, $P_p = (v_i, \dots, v_j)$ is contained in $F(P_k)$ and $B(P_{k-2}) \circ P_{k-1} \equiv B(P_k) \circ P_k^{-1}$ holds. Therefore, the condition 4 is satisfied for k . \square

(Case 3) Let $S = (v_0, \dots, v_l)$ be the spoke containing v and let i be the index such that $v_i = v$. Because v is not visited, $a(S) \leq i$ holds from the invariant 4. If $T(P(u) \circ e) \neq (v_0, \dots, v_i)$ holds, we return a path $P(u) \circ e$ (line 24). Otherwise, we set $P(v_j) \leftarrow (P(u) \circ e) \circ (v_i, \dots, v_j)$ and insert v_j into X for each index $j \in \{a(S), \dots, i-1\}$, and then update $a(S) \leftarrow i$ (lines 25–27).

Claim 3. $P(u) \circ e$ returned at line 24 is a y -augmenting path.

Proof. Because $T(P(u) \circ e) \neq (v_0, \dots, v_i) = B(t(P(u) \circ e))$ holds, the condition 2a of y -augment paths (Definition 3) is satisfied. From the invariant 4, for any segment P_k of $P(u) \circ e$ contained in the spoke S , $t(P(u) \circ e) = v_i$ is contained in $F(P_k)$. Therefore, the condition 2b is satisfied. \square

Claim 4. The lines 25–27 preserve all the invariants.

Proof. All the invariants excepting 1a are clearly preserved. We prove the invariant 1a by showing that $P(v_j)$ is a y -alternating path for any newly visited v_j . The conditions 1–3 of y -alternating paths (Definition 2) are clearly satisfied. Let $P_1 \circ \dots \circ P_p$ be the segments of $P(v_j)$. Because $B(P_{p-2}) \circ P_{p-1} = T(P(u) \circ e) \equiv (v_0, \dots, v_i) = B(P_p) \circ P_p^{-1}$ holds, the condition 4 is satisfied for p . Let P_k be a segment with $k < p$ contained in the same spoke S . From the invariant 4, P_k must be contained in $(v_{a(C)}, \dots, v_0)$. Therefore, P_p is contained in $F(P_k)$. Thus, the condition 4 is satisfied for k . \square

Now, we have finished proving Lemma 4 and the soundness of the algorithm. \square

Lemma 5. Any path returned by Algorithm 1 is a y -augmenting path, and any pair returned by the algorithm is a y -augmenting pair.

Implementation Detail

For achieving the linear-time complexity, we exploit the equivalence oracle as follows. For each integral cycle $C = (v_0, \dots, v_l)$ and for each index $i \in \{1, \dots, l-1\}$, we precompute $\mathcal{A}^*((v_0, \dots, v_i))$ and $\mathcal{A}^*((v_l, \dots, v_i))$. For each spoke $S = (v_0, \dots, v_l)$ and for each index $i \in \{1, \dots, l\}$, we precompute $\mathcal{A}^*((v_0, \dots, v_i))$. These precomputations can be done in $O(nT_E)$ time.

For each visited vertex v , instead of explicitly holding $P(v)$, we hold (1) $\text{prev}(v)$ that represents the edge picked at line 6 in the iteration when $P(v)$ is assigned and (2) $\text{tail}(v) := \mathcal{A}^*(T(P(v)))$. When the algorithm finds a y -augmenting path or pair, we restore $P(v)$ by using the table of $\text{prev}(v)$ in $O(n)$ time. For an edge $e = uv \in \delta(u) \setminus E(y)$, we have $\mathcal{A}^*(T(P(u)) \circ e) = \mathcal{A}(\text{tail}(u), e)$, which can be computed in $O(T_{\mathcal{E}})$ time. For any index j picked at line 15 or 25, we have $\text{tail}(v_j) = \mathcal{A}^*(T(P(v_j))) = \mathcal{A}^*((v_0, \dots, v_j))$, which has been precomputed. For any index j picked at line 19, we have $\text{tail}(v_j) = \mathcal{A}^*(T(P(v_j))) = \mathcal{A}^*((v_l, \dots, v_j))$, which has been precomputed. Thus, for any newly visited vertex w , we can compute $\text{tail}(w)$ in $O(T_{\mathcal{E}})$ time.

Finally, we show that each of the equivalence tests in the algorithm can be done in $O(T_{\mathcal{E}})$ time. Let $e = uv$ be the edge picked at line 6. First, we consider the equivalence test at line 8. If $\text{prev}(v) = e^{-1}$ holds, we have $T(P(v)) \equiv T(P(u)) \circ e$. Otherwise, $(T(P(v)), T(P(u)) \circ e)$ forms a single-branching pair from the invariant 2, and therefore, we can test $T(P(v)) \not\equiv T(P(u)) \circ e$ by asking $\mathcal{T}(\text{tail}(v), \mathcal{A}(\text{tail}(u), e))$ in $O(T_{\mathcal{E}})$ time. Next, we consider the equivalence test at line 14. Because $(T(P(u) \circ e), (v_l, \dots, v_i))$ forms a single-branching pair, we can test $T(P(u) \circ e) \not\equiv (v_l, \dots, v_i)$ by asking $\mathcal{T}(\mathcal{A}(\text{tail}(u), e), \mathcal{A}^*((v_l, \dots, v_i)))$. Because $\mathcal{A}^*((v_l, \dots, v_i))$ has been precomputed, this can be done in $O(T_{\mathcal{E}})$ time. The same argument applies to the equivalence tests at lines 18 and 24.

Now, we have shown that for any visited vertex v , $\text{tail}(v)$ can be computed in $O(T_{\mathcal{E}})$ time, and for any edge picked at line 6, all the equivalence tests can be done in $O(T_{\mathcal{E}})$ time. Because each vertex is pushed into X at most once and because each edge is processed at most twice (in both directions), we obtain the following lemma.

Lemma 6. *Algorithm 1 runs in $O(mT_{\mathcal{E}})$ time.*

3.3 Constructing Half-integral \mathcal{F} -Cover

In this subsection, we prove that if Algorithm 1 fails to find a y -augmenting path or pair, we can construct a half-integral \mathcal{F} -cover of the same size as follows. Let a and b be the tables used in Algorithm 1. First, we initialize $x(v) \leftarrow 0$ for all $v \in V$. For each integral cycle $C = (v_0, \dots, v_l)$, we set $x(v_{a(C)}) \leftarrow \frac{1}{2}$ and $x(v_{b(C)}) \leftarrow \frac{1}{2}$; when $a(C) = b(C)$, we set $x(v_{a(C)}) \leftarrow 1$. For each spoke $S = (v_0, \dots, v_l)$, we set $x(v_{a(S)}) \leftarrow \frac{1}{2}$. From the construction, we have $|x| = |y|$. We show that the function x is an \mathcal{F} -cover.

Lemma 7. *If Algorithm 1 fails to find a y -augmenting path or pair, the function x constructed as above is an \mathcal{F} -cover of size $|y|$.*

For proving this lemma, we use the following lemma.

Lemma 8. *For any s -walk Q with $x(V(Q)) = 0$, the vertex $t(Q)$ is visited and $T(P(t(Q))) \equiv Q$ holds.*

Proof. We prove the lemma by induction on the length of Q . The statement trivially holds for $Q = (s)$. Let $Q = Q' \circ uv$ be an s -walk with $x(V(Q)) = 0$. From the induction hypothesis, u is visited and $T(P(u)) \equiv Q'$ holds. We consider the following four cases: (Case 1) $v \notin V(y)$, (Case 2) $uv \notin E(y)$ and v is contained in an integral cycle, (Case 3) $uv \notin E(y)$ and v is contained in a spoke, or (Case 4) $uv \in E(y)$.

(Case 1) Consider the iteration when u is picked at line 5. If v is already visited, $T(P(v)) \equiv T(P(u)) \circ uv$ holds. Therefore, we have $T(P(v)) \equiv T(P(u)) \circ uv \equiv Q' \circ uv = Q$. If v is not visited yet, $P(v) = P(u) \circ uv$ holds. Therefore, we have $T(P(v)) = T(P(u)) \circ uv \equiv Q' \circ uv = Q$.

(Case 2) Let $C = (v_0, \dots, v_l)$ be the integral cycle containing v and let i be the index such that $v_i = v$. Consider the iteration when u is picked at line 5. Let $a(C)$ and $b(C)$ denote the values at the beginning of this iteration (hence, $x(v_{a(C)})$ and $x(v_{b(C)})$ might be zero). If $i < a(C)$ holds, v is already visited and $T(P(v)) = (v_0, \dots, v_i)$ holds. If $T(P(u)) \circ uv \not\equiv (v_0, \dots, v_i)$ additionally holds, the algorithm returns a y -augmenting pair at line 8. Therefore, we have $T(P(v)) = (v_0, \dots, v_i) \equiv T(P(u)) \circ uv \equiv Q' \circ uv = Q$. The same argument applies to the case of $i > b(C)$.

Now, we consider the remaining case that $a(C) \leq i \leq b(C)$ holds. Because $(v_0, \dots, v_i) \not\equiv (v_l, \dots, v_i)$ holds, at least one of $T(P(u) \circ uv) \not\equiv (v_0, \dots, v_i)$ or $T(P(u) \circ uv) \not\equiv (v_l, \dots, v_i)$ holds. If both of them hold, $a(C)$ and $b(C)$ are both set to i after this iteration; and therefore, we have $x(v) = 1$, which is a contradiction. If $T(P(u) \circ uv) \equiv (v_0, \dots, v_i)$ holds, $a(C)$ is set to i after this iteration. Because $x(v) = 0$, $a(C)$ must be greater than i at the end of the algorithm. Therefore, we have $T(P(v)) = (v_0, \dots, v_i) \equiv T(P(u) \circ uv) = T(P(u)) \circ uv \equiv Q' \circ uv = Q$. The same argument applies to the case of $T(P(u) \circ uv) \equiv (v_l, \dots, v_i)$.

(Case 3) Let $S = (v_0, \dots, v_l)$ be the spoke containing v and let i be the index such that $v_i = v$. Consider the iteration when u is picked at line 5. Let $a(S)$ denote the value at the beginning of this iteration. If $i < a(S)$ holds, v is already visited and $T(P(v)) = (v_0, \dots, v_i)$ holds. If $T(P(u)) \circ uv \not\equiv (v_0, \dots, v_i)$ additionally holds, the algorithm returns a y -augmenting pair at line 8. Therefore, we have $T(P(v)) = (v_0, \dots, v_i) \equiv T(P(u)) \circ uv \equiv Q' \circ uv = Q$.

Now, we consider the remaining case that $a(C) \leq i$ holds. If $T(P(u) \circ uv) \not\equiv (v_0, \dots, v_i)$ holds, the algorithm returns a y -augmenting path at line 24. Therefore, $T(P(u) \circ uv) \equiv (v_0, \dots, v_i)$ holds and $a(C)$ is set to i after this iteration. Because $x(v) = 0$, $a(C)$ must be greater than i at the end of the algorithm. Therefore, we have $T(P(v)) = (v_0, \dots, v_i) \equiv T(P(u) \circ uv) = T(P(u)) \circ uv \equiv Q' \circ uv = Q$.

(Case 4) Note that in this case, uv must be contained in an integral cycle or a spoke because otherwise, the algorithm returns a y -augmenting path at line 29. Let (v_0, \dots, v_l) be the integral cycle or the spoke containing uv and let i and j be the indices such that $v_i = v$ and $v_j = u$. Because u is visited, $T(P(u))$ is either (v_0, \dots, v_j) or (v_l, \dots, v_j) , and w.l.o.g., we can assume the former case. If $i = j - 1$, v is also visited and we have $T(P(v)) = (v_0, \dots, v_{j-1}) \equiv (v_0, \dots, v_j) \circ uv = T(P(u)) \circ uv \equiv Q' \circ uv = Q$. If $i = j + 1$ and v is not visited, we have $x(v) \geq \frac{1}{2}$, which is a contradiction. Therefore, v is also visited and we have $T(P(v)) = (v_0, \dots, v_{j+1}) = (v_0, \dots, v_j) \circ uv = T(P(u)) \circ uv \equiv Q' \circ uv = Q$. \square

Proof of Lemma 7. Suppose that there exists an s -cycle $C \in \mathcal{F}$ with $x(V(C)) < 1$. Because $x(v) \in \{0, \frac{1}{2}, 1\}$ holds, $x(V(C))$ is either 0 or $\frac{1}{2}$. If $x(V(C)) = 0$, by applying Lemma 8 against C , we have $C \equiv T(P(s)) = (s)$, which is a contradiction. If $x(V(C)) = \frac{1}{2}$, let t be the vertex with $x(t) = \frac{1}{2}$ on C , and let ut and vt be the two edges incident to t on C . Then, we can write $C = Q_1 \circ (u, t, v) \circ Q_2^{-1}$ for two s -walks Q_1 and Q_2 with $x(V(Q_1)) = x(V(Q_2)) = 0$. By applying Lemma 8 against Q_1 and Q_2 , we have $Q_1 \equiv T(P(u))$ and $Q_2 \equiv T(P(v))$. Therefore $T(P(u)) \circ ut \not\equiv T(P(v)) \circ vt$ holds.

Because $x(t) = \frac{1}{2}$, t is contained in an integral cycle or a spoke. Let $A = (v_0, \dots, v_l)$ be the integral cycle or the spoke containing t and let i be the index such that $v_i = t$. Because $x(t) = \frac{1}{2}$, exactly one of $a(A) = i$ or $b(A) = i$ holds, and w.l.o.g., we can assume $a(A) = i$.

Suppose that ut is contained in A . Because u is visited, we have $u = v_{i-1}$ and $T(P(u)) = (v_0, \dots, v_{i-1})$. Therefore, we have $T(P(u)) \circ ut = (v_0, \dots, v_i)$. The same argument applies to vt .

Suppose that ut is not contained in A and $T(P(u)) \circ ut \not\equiv (v_0, \dots, v_i)$ holds. Consider the iteration when u is picked at line 5. If A is an integral cycle, $b(A)$ is set to i after the iteration and therefore $x(t)$ is set to one, which is a contradiction. If A is a spoke, the algorithm returns a y -augmenting path at line 24, which is also a contradiction. Therefore, if ut is not contained in A , $T(P(u)) \circ ut \equiv (v_0, \dots, v_i)$ holds. The same argument applies to vt .

Now, we have proved that $T(P(u)) \circ ut \equiv (v_0, \dots, v_i) \equiv T(P(v)) \circ vt$ holds, which is a contradiction. \square

3.4 Augmentation

3.4.1 Simplification of Alternating Path

Before the proofs of Lemmas 2 and 3, we introduce a useful procedure to simplify an augmenting path/pair obtained by Algorithm 1. We first define such an operation in a formal way and prove the validity just after the definition.

Definition 5. For a basic \mathcal{F} -packing y and a y -alternating path $P = P_1 \circ \dots \circ P_l$ with $l \geq 2$, the *simplification* (y', P') of (y, P) is defined as follows (see Figure 5).

- A function y' is constructed from y as follows:
 - if P_2 is contained in an integral cycle C , replace C with an integral cycle $P_1 \circ F(P_2)^{-1}$;
 - if P_2 is contained in a spoke S , replace S with a spoke $P_1 \circ F(P_2)^{-1}$.
- A walk P' is defined as a concatenation $(B_y(P_2) \circ P_3) \circ P_4 \circ P_5 \circ \dots \circ P_l$ of $\max\{l - 2, 1\}$ paths¹³.

Lemma 9. For any basic \mathcal{F} -packing y and any y -alternating path $P = P_1 \circ \dots \circ P_l$ with $l \geq 2$, the simplification (y', P') of (y, P) satisfies the followings.

1. y' is a basic \mathcal{F} -packing with $|y'| = |y|$.
2. For any even $i \geq 4$, the following holds.
 - (a) If P_i is contained in an integral cycle in y , it is also contained in an integral cycle in y' ; moreover, $F_{y'}(P_i) \equiv F_y(P_i)$ and $B_{y'}(P_i) \equiv B_y(P_i)$ hold.
 - (b) If P_i is contained in a spoke in y , it is also contained in a spoke in y' in the direction towards s ; moreover, $F_{y'}(P_i) = F_y(P_i)$ and $B_{y'}(P_i) \equiv B_y(P_i)$ hold.
3. $P' = (B_y(P_2) \circ P_3) \circ P_4 \circ P_5 \circ \dots \circ P_l$ is a y' -alternating path, where $B_y(P_2) \circ P_3$ is the first segment and P_i ($i \geq 4$) is the $(i - 2)$ -th segment¹⁴.

Proof. First, we prove the first claim. When P_2 is contained in an integral cycle, from the condition 4a of y -alternating paths (Definition 2), $P_1 \not\equiv F(P_2)$ holds; i.e., $C' := P_1 \circ F(P_2)^{-1} \in \mathcal{F}$. Because P_1 is internally disjoint from $F(P_2)$, C' is a simple cycle. Thus, we can replace C with the integral cycle C' . When P_2 is contained in a spoke S_i of a wheel, from the condition 4b,

¹³When $l = 2$, $P' = B_y(P_2)$.

¹⁴When $l = 2$, P' consists of the single segment $B_y(P_2)$.

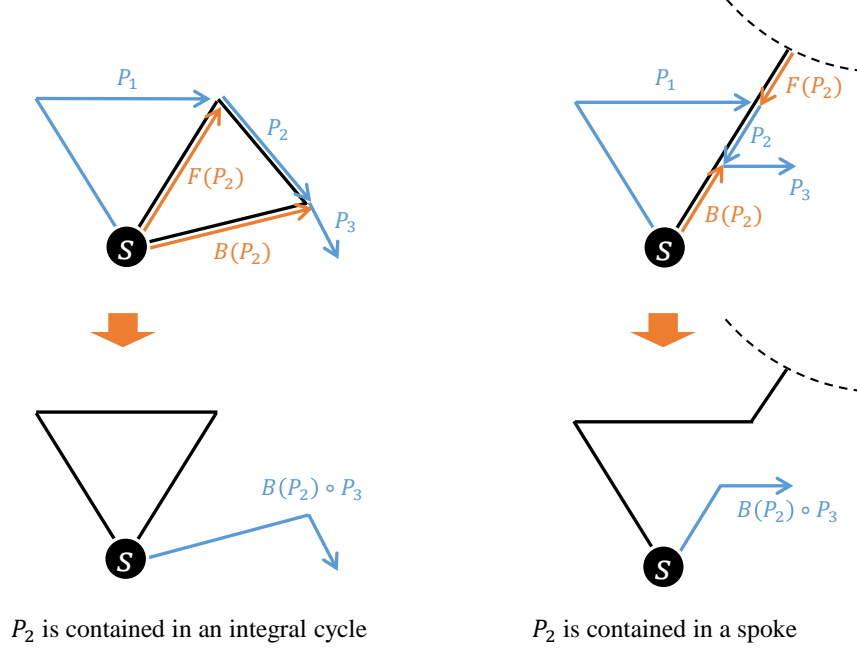


Figure 5: Simplification of a basic \mathcal{F} -packing and an alternating path.

$P_1 \equiv B(P_2) \circ P_2^{-1}$ holds. From the definition of the wheel (Definition 1), $S_{i-1} \circ H_{i-1} \circ S_i^{-1}$ and $S_i \circ H_i \circ S_{i+1}^{-1}$ are both in \mathcal{F} . As $B(P_2) \circ P_2^{-1} \circ F(P_2)^{-1} = S_i$ holds, we have $S'_i := P_1 \circ F(P_2)^{-1} \equiv S_i$. When the degree of the wheel is at least three, we have that $S_{i-1} \circ H_{i-1} \circ S_i^{-1}$ and $S'_i \circ H_i \circ S_{i+1}^{-1}$ are both in \mathcal{F} . When the degree of the wheel is one, we have that $S'_1 \circ H_1 \circ S_1^{-1}$ is in \mathcal{F} . Thus, this replacement of the spoke preserves the condition for the wheel.

Next, we prove the second claim. We can observe the following for any even $i \geq 4$.

- If both of P_2 and P_i are contained in the same integral cycle in y , from the condition 4a in Definition 2, P_i is contained in $F_y(P_2)$ and $P_1 \equiv B_y(P_2) \circ P_2^{-1}$ holds. Thus, P_i is contained in the integral cycle $P_1 \circ F_y(P_2)^{-1}$ in y' . If they have the same direction, P_2 is contained in $B_y(P_i)$, and we can write $B_y(P_i) = (B_y(P_2) \circ P_2^{-1}) \circ W$ for some subpath W . Then, it holds that $F_{y'}(P_i) = F_y(P_i)$ and $B_{y'}(P_i) = P_1 \circ W \equiv (B_y(P_2) \circ P_2^{-1}) \circ W = B_y(P_i)$. If they have the opposite direction, P_2 is contained in $F_y(P_i)$, and we can write $F_y(P_i) = (B_y(P_2) \circ P_2^{-1}) \circ W$ for some subpath W . Then, it holds that $F_{y'}(P_i) = P_1 \circ W \equiv (B_y(P_2) \circ P_2^{-1}) \circ W = F_y(P_i)$ and $B_{y'}(P_i) = B_y(P_i)$.
- If both of P_2 and P_i are contained in the same spoke in y , from the condition 4b in Definition 2, P_i is contained in $F_y(P_2)$ and $P_1 \equiv B_y(P_2) \circ P_2^{-1}$ holds. Thus, P_i is contained in the spoke $P_1 \circ F_y(P_2)^{-1}$ in y' . Because both of them are directed toward the root s , we can write $B_y(P_i) = (B_y(P_2) \circ P_2^{-1}) \circ W$ for some subpath W . Then, it holds that $F_{y'}(P_i) = F_y(P_i)$ and $B_{y'}(P_i) = P_1 \circ W \equiv (B_y(P_2) \circ P_2^{-1}) \circ W = B_y(P_i)$.
- Otherwise, the integral cycle or the spoke containing P_i does not change, and thus we have $F_{y'}(P_i) = F_y(P_i)$ and $B_{y'}(P_i) = B_y(P_i)$.

Finally, we prove the third claim. Because $B_y(P_2)$ does not contain any P_i , the conditions 1–3

in Definition 2 are satisfied. From the second claim and the property that $A \circ W \equiv B \circ W$ if $A \equiv B$, none of the three equivalence relations appeared in the condition 4 change. As we have seen in the proof of the second claim, $E(B_{y'}(P_i)) \setminus E(B_y(P_i)) \subseteq E(P_1)$ and $E(F_{y'}(P_i)) \setminus E(F_y(P_i)) \subseteq E(P_1)$ hold. Therefore, for any $i \geq 4$, none of the P_j 's with $j > i$ are newly contained in $B_{y'}(P_i)$ or $F_{y'}(P_i)$. Thus, the condition 4 is satisfied. \square

By applying the simplifying operation repeatedly, we obtain the following corollaries from Lemma 9.

Corollary 2. *Given a basic \mathcal{F} -packing y , a y -alternating path $P = P_1 \circ \dots \circ P_l$, and an even integer $l' \leq l$, a basic \mathcal{F} -packing y' of the same size and a y' -alternating path P' satisfying the following conditions can be constructed in linear time.*

1. P' can be written as $P' = (B' \circ P_{l'+1}) \circ P_{l'+2} \circ \dots \circ P_l$ for some s -path $B' \equiv B_y(P_{l'})$.
2. For any even $i \geq l' + 2$, the following holds.
 - (a) If P_i is contained in an integral cycle in y , it is also contained in an integral cycle in y' ; moreover, $F_{y'}(P_i) \equiv F_y(P_i)$ and $B_{y'}(P_i) \equiv B_y(P_i)$ hold.
 - (b) If P_i is contained in a spoke in y , it is also contained in a spoke in y' in the direction towards s ; moreover, $F_{y'}(P_i) = F_y(P_i)$ and $B_{y'}(P_i) \equiv B_y(P_i)$ hold.

Corollary 3. *Given a basic \mathcal{F} -packing y and a y -alternating path $P = P_1 \circ \dots \circ P_l$, a basic \mathcal{F} -packing y' of the same size and a single-segment y' -alternating path P' satisfying $P' \equiv T_y(P)$ can be constructed in linear time.*

3.4.2 Augmentation by Augmenting Path

The goal of this subsection is to prove Lemma 2.

Proof of Lemma 2. Let $P = P_1 \circ \dots \circ P_l$ be the given y -augmenting path and $t = t(P)$. If $l > 1$, by Corollary 3, we obtain in linear time a basic \mathcal{F} -packing y' of the same size and a single-segment y' -alternating path P' such that $t(P') = t$ and $P' \equiv T_y(P)$. We now show that P' is a y' -augmenting path. If P satisfies the first condition of y -augmenting paths (Definition 3), t is still contained in the same half-integral cycle in y' ; thus, P' is a y' -augmenting path. If P satisfies the second condition, the spoke S that contains t in y might not exist in y' ; however, from the condition 2b, t is still contained in a spoke S' (which may not be identical to S) in y' . By the same argument as in the proof of the second statement of Lemma 9, we have $B_{y'}(t) \equiv B_y(t)$. Therefore, we have $T_{y'}(P') = P' \equiv T_y(P) \not\equiv B_y(t) \equiv B_{y'}(t)$. Thus, P' is a y' -augmenting path.

Now, we can concentrate on the case when $l = 1$ (see Figure 6). Let $(\{S_1, \dots, S_d\}, H_1 \circ \dots \circ H_d)$ be the wheel containing t . W.l.o.g., we can assume that t is contained in H_d or S_d .

If t is contained in H_d , let F be the prefix subpath of H_d to t and B be the suffix subpath of H_d from t ; i.e., $F \circ B = H_d$. Because $S_d \circ H_d \circ S_1^{-1} \in \mathcal{F}$, we have $S_d \circ F \not\equiv S_1 \circ B^{-1}$, and thus we have $S_d \circ F \circ P^{-1} \not\equiv S_1 \circ B^{-1} \circ P^{-1}$. Therefore, at least one of $S_d \circ F \circ P^{-1}$ and $S_1 \circ B^{-1} \circ P^{-1}$ is in \mathcal{F} , and w.l.o.g., we can assume the former case. Then, by decomposing the wheel into $(d-1)/2$ integral cycles $\{S_1 \circ H_1 \circ S_2^{-1}, S_3 \circ H_3 \circ S_4^{-1}, \dots, S_{d-2} \circ H_{d-2} \circ S_{d-1}^{-1}\}$ and by inserting an integral cycle $S_d \circ F \circ P^{-1}$, we can obtain a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$.

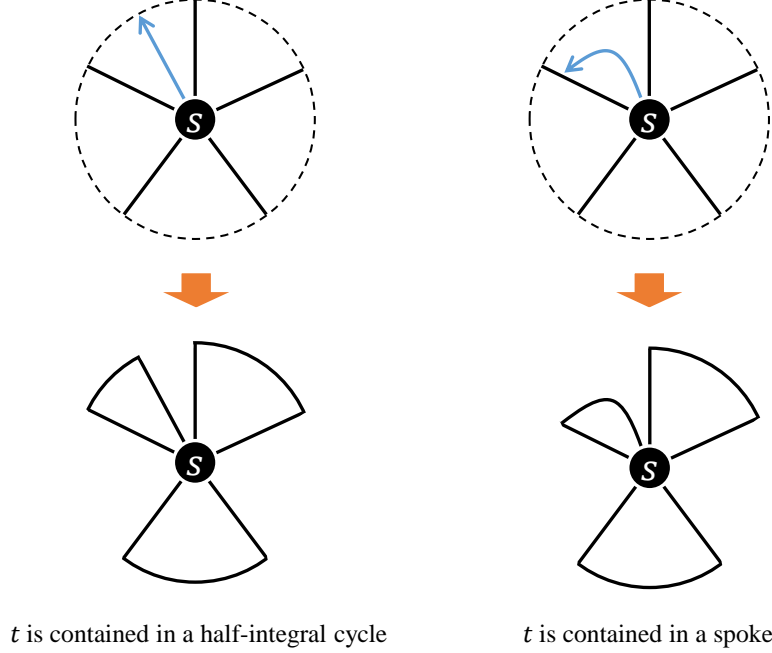


Figure 6: Augmentation by a single-segment augmenting path.

If t is contained in S_d , we have $P \neq B(t)$. Then, by decomposing the wheel into $(d-1)/2$ integral cycles $\{S_1 \circ H_1 \circ S_2^{-1}, S_3 \circ H_3 \circ S_4^{-1}, \dots, S_{d-2} \circ H_{d-2} \circ S_{d-1}^{-1}\}$ and by inserting an integral cycle $P \circ B(t)^{-1}$, we can obtain a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$. \square

3.4.3 Augmentation by Augmenting Pair

The goal of this subsection is to prove Lemma 3.

Intuitively, we want to augment y as follows. If P and Q share no edges, we first simplify P and Q independently by applying Corollary 3, and finally we obtain a new integral cycle $P' \circ Q'^{-1}$. If P and Q share edges, we first simplify the common prefix R of (P, Q) by applying Corollary 2, and finally we obtain a new wheel whose half-integral cycle is $P' \circ Q'^{-1}$. However, it is not that easy; when P' or Q' intersects with spokes or intersects with the same integral cycle multiple times, this approach does not work. For this reason, we augment y by gradually simplifying the augmenting pair. First, we prove the lemma against a special case.

Lemma 10. *Given a basic \mathcal{F} -packing y and a y -augmenting pair $(P = P_1 \circ \dots \circ P_p, Q = Q_1 \circ \dots \circ Q_q)$ satisfying all the following conditions, a basic \mathcal{F} -packing of size at least $|y| + \frac{1}{2}$ can be constructed in linear time.*

1. None of the P_i 's are contained in the spokes.
2. None of the integral cycles contain multiple P_i 's.
3. One of the following two conditions is satisfied:
 - (a) $p \geq q = 1$ or

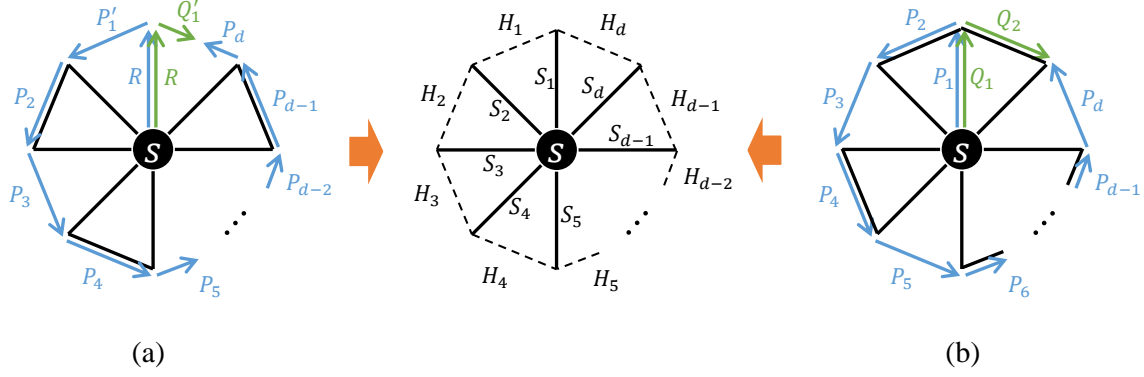


Figure 7: Applying Lemma 10.

(b) $p \geq q = 2$, $P_1 = Q_1$, and P_2 and Q_2 are contained in an integral cycle in the opposite direction.

Proof. When (P, Q) satisfies the condition 3a, we further divide the case into the following three cases: (1) P and Q share no edges, (2) they share some edges and $p = 1$, or (3) they share some edges and $p \geq 2$.

In the first case, we first apply Corollary 3 against P and obtain a basic \mathcal{F} -packing y' of the same size and a single-segment y' -alternating path P' satisfying $P' \equiv T_{y'}(P) \not\equiv T_{y'}(Q) = Q$. From the construction of P' , P' and Q share no edges. Therefore, we can obtain a basic \mathcal{F} -packing of size $|y| + 1$ by introducing a new integral cycle $P' \circ Q^{-1} \in \mathcal{F}$.

In the second case, (P, Q) is a single-branching pair such that $P \not\equiv Q$. Therefore, we can obtain a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$ by introducing a new wheel $P \circ Q^{-1}$ of degree one.

In the third case, let R be the common prefix and let us write $P_1 = R \circ P'_1$ and $Q_1 = R \circ Q'_1$. If p is even, we set $d := p + 1$ and $P_d := (t(P))$, and otherwise, we set $d := p$. We define paths $\{H_1, \dots, H_d\}$ and $\{S_1, \dots, S_d\}$ as follows (see Figure 7 (a)).

- $H_1 := P'_1$.
- $H_i := P_i$ for $i \in \{2, \dots, d-1\}$.
- $H_d := P_d \circ Q'^{-1}_1$.
- $S_1 := R$.
- $S_i := F(P_i)$ for even $i \in \{2, 4, \dots, d-1\}$.
- $S_i := B(P_{i-1})$ for odd $i \in \{3, 5, \dots, d\}$.

Now, we show that these paths form a wheel. The first two conditions of the wheel (Definition 1) are trivially satisfied. As none of the integral cycles in y contain multiple P_i 's, the third and the fourth conditions are satisfied. We can see that the fifth condition is satisfied as follows.

- $S_1 \circ H_1 \circ S_2^{-1} = R \circ P'_1 \circ F(P_2)^{-1} = P_1 \circ F(P_2)^{-1} \in \mathcal{F}$.
- $S_i \circ H_i \circ S_{i+1}^{-1} = F(P_i) \circ P_i \circ B(P_i)^{-1} \in \mathcal{F}$ for even $i \in \{2, 4, \dots, d-1\}$.

- $S_i \circ H_i \circ S_{i+1}^{-1} = B(P_{i-1}) \circ P_i \circ F(P_{i+1})^{-1} \in \mathcal{F}$ for odd $i \in \{3, 5, \dots, d-2\}$.
- $S_d \circ H_d \circ S_1^{-1} = B(P_{d-1}) \circ P_d \circ Q_1'^{-1} \circ R^{-1} = T(P) \circ T(Q)^{-1} \in \mathcal{F}$.

Thus, by removing the $(d-1)/2$ integral cycles intersecting P and by inserting the wheel of degree d , we can obtain a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$.

Finally, we consider the case when (P, Q) satisfies the condition 3b. Note that p must be odd from the condition 3 of y -augmenting pairs (Definition 4). Let $d := p$. We define paths $\{H_1, \dots, H_d\}$ and $\{S_1, \dots, S_d\}$ as follows (see Figure 7 (b)).

- $H_i := P_{i+1}$ for $i \in \{1, \dots, d-1\}$.
- $H_d := Q_2^{-1}$.
- $S_1 := P_1$.
- $S_i := B(P_i)$ for even $i \in \{2, 4, \dots, d-1\}$.
- $S_i := F(P_{i+1})$ for odd $i \in \{3, 5, \dots, d-2\}$.
- $S_d := B(Q_2)$.

Now, we show that these paths form a wheel. The first two conditions of the wheel are trivially satisfied. Because none of the integral cycles in y contain multiple P_i 's and because $s(P_2) = s(Q_2) = t(Q_2^{-1})$, the third and the fourth conditions are satisfied. We can see that the fifth condition is satisfied as follows.

- $S_1 \circ H_1 \circ S_2^{-1} = P_1 \circ P_2 \circ B(P_2)^{-1} = P_1 \circ (B(P_2) \circ P_2^{-1})^{-1} = Q_1 \circ F(Q_2)^{-1} \in \mathcal{F}$.
- $S_i \circ H_i \circ S_{i+1}^{-1} = B(P_i) \circ P_{i+1} \circ F(P_{i+2})^{-1} \in \mathcal{F}$ for even $i \in \{2, 4, \dots, d-3\}$.
- $S_i \circ H_i \circ S_{i+1}^{-1} = F(P_{i+1}) \circ P_{i+1} \circ B(P_{i+1})^{-1} \in \mathcal{F}$ for odd $i \in \{3, 5, \dots, d-2\}$.
- $S_{d-1} \circ H_{d-1} \circ S_d^{-1} = B(P_{d-1}) \circ P_d \circ B(Q_2)^{-1} = T(P) \circ T(Q)^{-1} \in \mathcal{F}$.
- $S_d \circ H_d \circ S_1^{-1} = B(Q_2) \circ Q_2^{-1} \circ P_1^{-1} = F(P_2) \circ P_1^{-1} \in \mathcal{F}$.

Thus, by removing the $(d-1)/2$ integral cycles intersecting P and by inserting the wheel of degree d , we can obtain a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$. \square

Next, we give two lemmas for weakening the assumptions in the above lemma.

Lemma 11. *Given a basic \mathcal{F} -packing y and a y -augmenting pair $(P = P_1 \circ \dots \circ P_p, Q = Q_1 \circ \dots \circ Q_q)$ satisfying all the following conditions, a basic \mathcal{F} -packing of size at least $|y| + \frac{1}{2}$ can be constructed in linear time.*

1. None of the P_i 's are contained in the spokes.
- 2'. Any two segments of P contained in the same integral cycle have the same direction.
3. One of the following two conditions is satisfied:
 - (a) $p \geq q = 1$ or

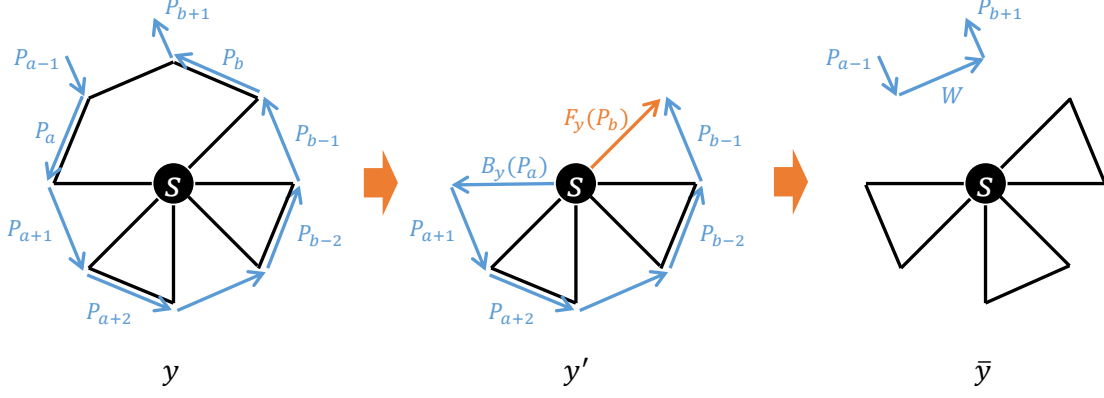


Figure 8: Applying Lemma 11.

- (b) $p \geq q = 2$, $P_1 = Q_1$, and P_2 and Q_2 are contained in an integral cycle in the opposite direction.

Proof. Note that the conditions 1 and 3 are same as those in Lemma 10, and if the condition 3b is satisfied, from the condition 4a of y -alternating paths (Definition 2), the integral cycle containing P_2 and Q_2 never contain any other segments. We call a segment of P *obstructive* if it is contained in an integral cycle containing multiple segments of P . If there exist no obstructive segments, the condition 2 of Lemma 10 is satisfied; thus, we can obtain a basic \mathcal{F} -packing of size at least $|y| + \frac{1}{2}$ by applying Lemma 10. We repeat the following process while there exist obstructive segments.

Let P_a be the first obstructive segment and let P_b ($b > a$) be the next segment contained in the integral cycle containing P_a . From the condition 4a of y -alternating paths (Definition 2), P_b is contained in $F(P_a)$ and $B(P_{a-2}) \circ P_{a-1} \equiv B(P_a) \circ P_a^{-1}$ holds. We construct a basic \mathcal{F} -packing \bar{y} of the same size and a \bar{y} -augmenting pair (\bar{P}, Q) such that the precondition of this lemma is satisfied and the number of obstructive segments strictly decreases as follows (see Figure 8).

First, we construct a basic \mathcal{F} -packing y' of size $|y| - 1$ by removing the integral cycle containing P_a and P_b from y . Observe that $P' := (B_y(P_a) \circ P_{a+1}) \circ P_{a+2} \circ \dots \circ P_{b-1}$ is a y' -alternating path. Then, by applying Corollary 3 against y' and P' , we obtain a basic \mathcal{F} -packing y'' of size $|y| - 1$ and a single-segment y'' -alternating path P'' satisfying $P'' \equiv T_{y'}(P') = B_{y'}(P_{b-2}) \circ P_{b-1} = B_y(P_{b-2}) \circ P_{b-1}$. Here, from the condition 4a of the y -alternating path P , $B_y(P_{b-2}) \circ P_{b-1} \not\equiv F_y(P_b)$ holds, and from the choice of P_b , $F_y(P_b)$ is internally disjoint from P'' . Thus, we can obtain a basic \mathcal{F} -packing \bar{y} of size $|y|$ by introducing a new integral cycle $P'' \circ F_y(P_b)$. Let W be the path from $s(P_a)$ to $t(P_b)$ along the integral cycle and let $\bar{P} := P_1 \circ \dots \circ P_{a-2} \circ (P_{a-1} \circ W \circ P_{b+1}) \circ P_{b+2} \circ \dots \circ P_p$. Finally, we prove that (\bar{P}, Q) is a \bar{y} -augmenting pair satisfying the preconditions of this lemma. From the construction of \bar{y} and \bar{P} , the conditions 1-3 of \bar{y} -alternating paths (Definition 2) are clearly satisfied. Because $B_{\bar{y}}(P_{a-2}) \circ (P_{a-1} \circ W \circ P_{b+1}) \equiv B_y(P_{a-2}) \circ P_{a-1} \circ W \circ P_{b+1} \equiv B_y(P_a) \circ P_a^{-1} \circ W \circ P_{b+1} \equiv B_y(P_b) \circ P_{b+1}$ holds, the condition 4a is satisfied. Thus, \bar{P} is a \bar{y} -alternating path. Because $T_{\bar{y}}(\bar{P}) \equiv T_y(P) \neq T_y(Q) \equiv T_{\bar{y}}(Q)$ holds and because (\bar{P}, Q) satisfies the precondition 3 of the lemma, (\bar{P}, Q) is a \bar{y} -augmenting pair. Because no segments of \bar{P} are newly contained in the spokes in \bar{y} , the condition 1 of the lemma is satisfied. From the choice of P_a , for any even $i \in \{a+2, \dots, b-2\}$, $B_{y'}(P_i)$ contains no segments from $\{P_2, \dots, P_{a-2}\}$. Therefore, no two segments of \bar{P} are newly contained in a same integral cycle in \bar{y} . Thus, the condition 2' is satisfied. When (P, Q) satisfies the condition 3a, (\bar{P}, Q) also satisfies the condition 3a. When (P, Q) satisfies the condition 3b, (\bar{P}, Q) also satisfies

the condition 3b because the integral cycle containing P_2 and Q_2 remains in \bar{y} . Thus, all the conditions in the lemma are satisfied.

We can find the pair (P_a, P_b) by gradually increasing an index i , which is not reset during the repetition, and searching for P_j contained in $F(P_i)$ by traversing the integral cycle. Therefore, each edge is traversed at most once through the whole process, and thus, the total running time is linear in the graph size. \square

Lemma 12. *Given a basic \mathcal{F} -packing y and a y -augmenting pair $(P = P_1 \circ \dots \circ P_p, Q = Q_1 \circ \dots \circ Q_q)$, either of a y -augmenting path or a y -augmenting pair (\bar{P}, \bar{Q}) satisfying all the following conditions can be constructed in linear time.*

1. All the segments of \bar{Q} excepting the last one are contained in \bar{P} .
2. \bar{P} can be written as $\bar{P} = P \circ Q_q^{-1} \circ Q_{q-1}^{-1} \circ \dots \circ Q_{q'}^{-1}$ for some q' .
3. The common prefix of (\bar{P}, \bar{Q}) contains the common prefix of (P, Q) .
4. The following two conditions are satisfied for the new segments $S := \{Q_q^{-1}, \dots, Q_{q'}^{-1}\}$ of \bar{P} :
 - (a) no segments in S are contained in the spokes and
 - (b) any two segments in S contained in the same integral cycle have the same direction.

Proof. Initially, the conditions 2–4 are trivially satisfied. We repeat the following process by preserving these conditions, and when q becomes one or Q_{q-1} gets contained in P , the condition 1 is satisfied.

(Case 1) If both of p and q are odd, we update $P' \leftarrow P_1 \circ \dots \circ P_{p-1} \circ (P_p \circ Q_q^{-1})$ and $Q' \leftarrow Q_1 \circ \dots \circ Q_{q-1}$. Because P_p and Q_q share no edges and because $T(P') \circ T(Q')^{-1} = (B(P_{p-1}) \circ (P_p \circ Q_q^{-1})) \circ B(Q_{q-1})^{-1} = (B(P_{p-1}) \circ P_p) \circ (B(Q_{q-1}) \circ Q_q)^{-1} = T(P) \circ T(Q)^{-1}$ holds, (P', Q') is a y -augmenting pair.

(Case 2) If p is even and q is odd, we update $P' \leftarrow P_1 \circ \dots \circ P_{p-1} \circ P_p \circ Q_q^{-1}$ and $Q' \leftarrow Q_1 \circ \dots \circ Q_{q-1}$. Because P_p and Q_q share no edges and because $T(P') \circ T(Q')^{-1} = (B(P_p) \circ Q_q^{-1}) \circ B(Q_{q-1})^{-1} = B(P_p) \circ (B(Q_{q-1}) \circ Q_q)^{-1} = T(P) \circ T(Q)^{-1}$ holds, (P', Q') is a y -augmenting pair.

(Case 3) If p is odd, q is even, and if Q_q is contained in a spoke, we search for the nearest segment of P contained in $F(Q_q)$ by traversing the spoke. Note that none of the other segments of Q are contained in $F(Q_q)$ from the condition 4b of y -alternating paths (Definition 2), and each segment of P is fully contained in $F(Q_q)$ or internally disjoint from $F(Q_q)$ because Q_q shares no edges with P ; thus, each edge is traversed at most once through the entire process, and therefore the total running time of this part is linear in the graph size. If none of the P_i 's are contained in $F(Q_q)$, P is a y -augmenting path because $t(P)$ is contained in the spoke and $T(P) \neq T(Q) = B(t(P))$ holds. Otherwise, let P_k be the nearest segment of P contained in $F(Q_q)$, let W be the path from $s(P_k)$ to $t(P)$ along the spoke, and let $\bar{Q} := P_1 \circ \dots \circ P_{k-1} \circ W$ (see Figure 9 (left)). Because $T(\bar{Q}) = B(W) = B(Q_q) = T(Q)$ holds, (P, \bar{Q}) is a y -augmenting pair. Note that this finishes the process.

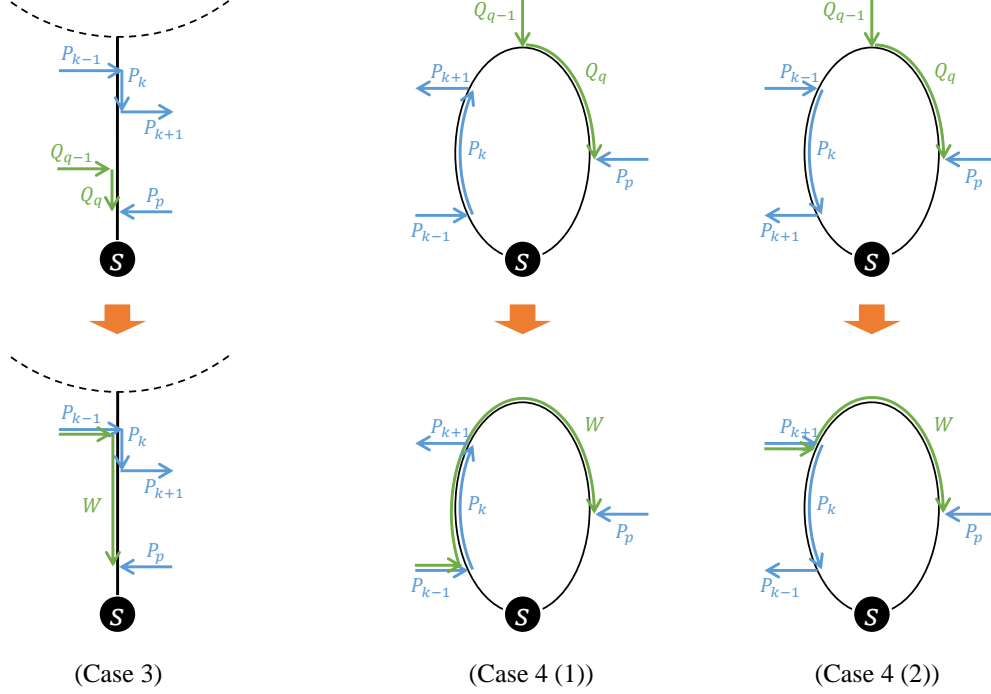


Figure 9: Applying Lemma 12.

(Case 4) If p is odd, q is even, and if Q_q is contained in an integral cycle, we search for the nearest segment of P or Q contained in $F(Q_q)$ by traversing the integral cycle. Because each edge is traversed at most twice (in two directions) through the entire process, the total running time of this part is linear in the graph size. Note that from the condition 4a of the y -alternating path Q (Definition 2), $F(Q_q)$ cannot contain a segment Q_k such that Q_k has the same direction as Q_q or Q_k has the opposite direction as Q_q and $B(Q_{k-2}) \circ Q_{k-1} \neq B(Q_k) \circ Q_k^{-1}$ holds.

If the nearest segment is P_k such that (1) P_k has the same direction as Q_q or (2) P_k has the opposite direction as Q_q and $B(P_{k-2}) \circ P_{k-1} \neq B(P_k) \circ P_k^{-1}$ holds, let W be the path from $s(P_k)$ to $t(P)$ along the integral cycle and let $\bar{Q} := P_1 \circ \dots \circ P_{k-1} \circ W$ (see Figure 9 (right)). Because $T(\bar{Q}) = B(W) = B(Q_q) = T(Q)$ holds, (P, \bar{Q}) is a y -augmenting pair and we finish the process.

Otherwise (i.e., (a) no segments are contained in $F(Q_q)$, (b) the nearest segment is Q_k that has the opposite direction as Q_q and $B(Q_{k-2}) \circ Q_{k-1} \equiv B(Q_k) \circ Q_k^{-1}$ holds, or (c) the nearest segment is P_k that has the opposite direction as Q_q and $B(P_{k-2}) \circ P_{k-1} \equiv B(P_k) \circ P_k^{-1}$ holds), we update $P' \leftarrow P_1 \circ \dots \circ P_p \circ Q_q^{-1}$ and $Q' \leftarrow Q_1 \circ \dots \circ Q_{q-1}$. Note that in this case, from the condition 4a of y -alternating paths (Definition 2) and the condition 4 of y -augmenting pairs (Definition 4), the conditions (b) and (c) hold not only against the nearest segment Q_k or P_k but also against any segments contained in $F(Q_q)$.

First, we prove that P' is a y -alternating path. The first three conditions of y -alternating paths (Definition 2) are clearly satisfied. Because $B(P_{p-1}) \circ P_p = T(P) \neq T(Q) = B(Q_q) = F(Q_q^{-1})$ holds, the condition 4 is satisfied for $i = p + 1$. For checking the condition 4 against the other P_i 's, it suffices to show that, for any segment P_i contained in the same integral cycle as Q_q , it holds that $B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}$ and Q_q is contained in $F(P_i)$. Let P_i be a segment contained in $B(Q_q)$. From the condition 4 of y -augmenting pairs (Definition 4), P_i has the same direction as

Q_q . Thus, Q_q is contained in $F(P_i)$, and therefore, from the condition 4 again, $B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}$ holds. Let P_i be a segment contained in $F(Q_q)$. Then, as we have discussed above, $B(P_{i-2}) \circ P_{i-1} \equiv B(P_i) \circ P_i^{-1}$ holds and P_i has the opposite direction as Q_q , which means that Q_q is contained in $F(P_i)$.

Next, we prove that (P', Q') is a y -augmenting pair. The conditions 1 and 3 of y -augmenting pairs (Definition 4) are clearly satisfied. Because $T(P') = B(Q_q^{-1}) = F(Q_q) \neq B(Q_{q-2}) \circ Q_{q-1} = T(Q')$ holds, the condition 2 is satisfied. For checking the condition 4, it suffices to show that, (1) none of the Q_j 's with $j < q$ are contained in $B(Q_q^{-1}) = F(Q_q)$ in the same direction as Q_q , (2) if $B(P_{p-1}) \circ P_p \neq B(Q_q^{-1}) \circ (Q_q^{-1})^{-1} = F(Q_q) \circ Q_q$, none of the Q_j 's with $j < q$ are contained in $F(Q_q^{-1}) = B(Q_q)$ in the opposite direction as Q_q , (3) for any Q_j contained in an integral cycle, Q_q is not contained in $B(Q_j)$ in the same direction as Q_j , and (4) moreover if $B(Q_{j-2}) \circ Q_{j-1} \neq B(Q_j) \circ Q_j^{-1}$ holds, Q_q is not contained in $F(Q_j)$ in the opposite direction as Q_j . All of these conditions directly follow from the condition 4a of the y -alternating path Q (Definition 2).

Finally, we prove that (P', Q') satisfies the conditions 2–4 of this lemma. The conditions 2 and 3 are clearly satisfied. From the condition 4a of the initial y -alternating path Q , $B(Q_q)$ contains none of the new segments S . Thus, if a segment in S is contained in the same integral cycle as Q_q , it must be contained in $F(Q_q)$, and therefore, it has the same direction as Q_q^{-1} . \square

Finally, we prove Lemma 3 by combining Lemma 11 and 12.

Proof of Lemma 3. First, we apply Lemma 12 against (Q, P) . If we obtain a y -augmenting path, by applying Lemma 2, we obtain a basic \mathcal{F} -packing of size $|y| + \frac{1}{2}$. Otherwise, we obtain an updated y -augmenting pair (\bar{Q}, \bar{P}) such that all the segments of \bar{P} excepting the last one are contained in the common prefix.

Next, we apply Lemma 12 against (\bar{P}, \bar{Q}) . If we obtain a y -augmenting path, we finish, and otherwise, we obtain an updated y -augmenting pair (\hat{P}, \hat{Q}) . Let $\hat{P} := \hat{P}_1 \circ \dots \circ \hat{P}_{\hat{p}}$ and $\hat{Q} := \hat{Q}_1 \circ \dots \circ \hat{Q}_{\hat{q}}$. Then, $\hat{Q}_{\hat{q}-1}$ is contained in $\hat{P}_{\hat{q}-1}$. Note that this implies that $\hat{P}_i = \hat{Q}_i$ holds for any $i \in \{1, \dots, \hat{q} - 2\}$, and moreover, if \hat{q} is even, $\hat{P}_{\hat{q}-1} = \hat{Q}_{\hat{q}-1}$ also holds. Because the common prefix of (\hat{P}, \hat{Q}) contains the common prefix of (\bar{P}, \bar{Q}) , and because $\hat{P}_{\bar{p}-1}$ is contained in the common prefix of (\bar{P}, \bar{Q}) , we have $\hat{q} \geq \bar{p} - 1$. Let assume that $\hat{q} = \bar{p} - 1$ holds and \hat{q} is odd. In this case, $\hat{P}_{\hat{q}}$ is contained in $\hat{Q}_{\hat{q}}$, and therefore $\hat{P}_{\hat{q}} = \hat{Q}_{\hat{q}}$ holds. Then, from the condition 3 of the y -augmenting pair (\hat{P}, \hat{Q}) (Definition 4), \hat{p} must be even and $\hat{P}_{\hat{p}}$ must be contained in the same integral cycle as $\hat{P}_{\bar{p}}$; however, because $B(\hat{P}_{\bar{p}-2}) \circ \hat{P}_{\bar{p}-1} = T(\hat{Q}) \neq T(\hat{P}) = B(\hat{P}_{\bar{p}}) = B(\hat{P}_{\bar{p}}) \circ \hat{P}_{\bar{p}}^{-1}$ holds, this violates the condition 4a of the y -alternating path \hat{P} . Thus, if \hat{q} is odd, $\hat{q} \geq \bar{p}$ holds. We consider two cases.

If \hat{q} is odd or $\hat{P}_{\hat{q}}$ and $\hat{Q}_{\hat{q}}$ have the same direction, let l be the maximum even integer at most \hat{q} . We apply Corollary 2 against \hat{P} and \hat{Q} , independently, and obtain a basic \mathcal{F} -packing y' of the same size and y' -alternating paths P' and Q' . Here, because $\hat{P}_i = \hat{Q}_i$ holds for any $i \in \{1, \dots, l-1\}$ and because $F(\hat{P}_l) = F(\hat{Q}_l)$ holds, the basic \mathcal{F} -packing obtained by these two applications are the same; thus, we use the same symbol y' . Because $T_{y'}(P') \equiv T_y(\hat{P}) \neq T_y(\hat{Q}) \equiv T_{y'}(Q')$ holds, (P', Q') is a y' -augmenting pair. Now, we show that (P', Q') satisfies the conditions in Lemma 11. Because Q' is single-segment, the third condition is satisfied. When \hat{q} is odd, we have $l + 2 = \hat{q} + 1 > \bar{p}$, and when \hat{q} is even, we have $l + 2 = \hat{q} + 2 > \bar{p}$. Therefore, from the condition 4 of Lemma 12, the first and the second conditions hold. Thus, by applying Lemma 11 against (P', Q') , we obtain a basic \mathcal{F} -packing of size at least $|y| + \frac{1}{2}$.

If \hat{q} is even and $\hat{P}_{\hat{q}}$ and $\hat{Q}_{\hat{q}}$ have the opposite direction, let $l := \hat{q} - 2$. We apply Corollary 2 against \hat{P} and \hat{Q} , independently, and obtain a basic \mathcal{F} -packing y' of the same size and y' -alternating paths P' and Q' . Here, because $\hat{P}_i = \hat{Q}_i$ holds for any $i \in \{1, \dots, l\}$, the basic \mathcal{F} -packing and obtained by these two applications are the same; thus, we use the same symbol y' . Because $T_{y'}(P') \equiv T_y(\hat{P}) \neq T_y(\hat{Q}) \equiv T_{y'}(Q')$ holds, (P', Q') is a y' -augmenting pair. Now, we show that (P', Q') satisfies the conditions in Lemma 11. Because we set $l := \hat{q} - 2$, the number of segments of Q' is two. Because $\hat{P}_{\hat{q}-1} = \hat{Q}_{\hat{q}-1}$ holds, the first segments of P' and Q' are the same. Because $\hat{P}_{\hat{q}}$ and $\hat{Q}_{\hat{q}}$ have the opposite direction, the second segments of P' and Q' have the opposite direction. Thus, the third condition is satisfied. Note that in this case, $\hat{P}_{\hat{q}}$ and $\hat{Q}_{\hat{q}}$ are contained in an integral cycle and, from the condition 4a of y -alternating paths (Definition 2), the integral cycle containing $\hat{P}_{\hat{q}}$ and $\hat{Q}_{\hat{q}}$ never contain any other segments. Moreover, because $l + 4 = \hat{q} + 2 > \bar{p}$, the first and the second conditions follow from the condition 4 of Lemma 12. Thus, by applying Lemma 11 against (P', Q') , we obtain a basic \mathcal{F} -packing of size at least $|y| + \frac{1}{2}$. \square

4 Linear-Time FPT Algorithms

4.1 Half-Integral LPs and Implementations of Equivalence Oracles

We describe the half-integral LPs used in the branch-and-bound FPT algorithms for the problems considered in this paper. Refer to [18] or [35] for the derivation of these LPs. All the LPs excepting that for NODE MULTIWAY CUT are defined in the form of \mathcal{F} -cover¹⁵. For NODE MULTIWAY CUT, by inserting a new vertex s connected to all the terminals and by splitting each terminal t into $k+1$ vertices $\{t_0, \dots, t_k\}$ (to make sure that the terminals will not be deleted)¹⁶, the LP is reduced to the \mathcal{F} -covering. We show that each \mathcal{F} is nice by giving a definition of the equivalence relation (\equiv). We also give efficient implementations of the equivalence oracle.

LP for FEEDBACK VERTEX SET

Input: A graph $G = (V, E)$ and a root $s \in V$.

\mathcal{F} : $C \in \mathcal{F}$ iff $\mathbf{1}_{E(C)}(e)$ is odd for some edge $e \in E$.

(\equiv): $P \equiv Q$ iff $\mathbf{1}_{E(P)}(e) \equiv \mathbf{1}_{E(Q)}(e) \pmod{2}$ for all $e \in E$

Oracle: $P \not\equiv Q$ for any single-branching pair (P, Q) . Thus, both \mathcal{A} and \mathcal{T} take a constant time.

LP for GROUP FEEDBACK VERTEX SET

Input: A group $\Gamma = (D, \cdot)$ given as an $O(T_\Gamma)$ -time oracle performing the group operation (\cdot) , a Γ -labeled graph $G = (V, E)$ with labeling $\lambda : \hat{E} \rightarrow D$ with $\lambda(uv) \cdot \lambda(vu) = 1_\Gamma$ for every $uv \in \hat{E}$, where 1_Γ is the unity of Γ , and a root $s \in V$.

\mathcal{F} : $(v_0, \dots, v_l) \in \mathcal{F}$ iff $\lambda(v_0 v_1) \cdots \lambda(v_{l-1} v_l) \neq 1_\Gamma$.

¹⁵ To be precise, in [18], the following form of discrete relaxations are used instead of LPs: given a graph G , a root s , and a set \mathcal{F} of forbidden cycles passing through s , find disjoint sets of vertices V_1 and $V_{\frac{1}{2}}$ not containing s that minimize $|V_1| + \frac{1}{2}|V_{\frac{1}{2}}|$ under the constraints that any cycle $C \in \mathcal{F}$ passes at least one vertex in V_1 or at least two vertices in $V_{\frac{1}{2}}$. We can easily see that this is equivalent to the half-integral \mathcal{F} -cover by the following correspondence: $x(v) = 1$ iff $v \in V_1$, $x(v) = \frac{1}{2}$ iff $v \in V_{\frac{1}{2}}$, and $x(v) = 0$ iff $v \notin V_1 \cup V_{\frac{1}{2}}$.

¹⁶ When the number of terminals is large, this increases the graph size to $O(km)$. For efficiency, we can use the following reduction to \mathcal{F} -packing. First, we insert a new vertex s . Then, for each terminal t with neighbors $\{v_1, \dots, v_d\}$, we split t into d vertices $\{t_1, \dots, t_d\}$ and insert edges st_i and $t_i v_i$ for each $i \in \{1, \dots, d\}$. The number of edges in this graph is $O(m)$.

(\equiv) : $(v_0, \dots, v_p) \equiv (u_0, \dots, u_q)$ iff $\lambda(v_0v_1) \cdots \lambda(v_{p-1}v_p) = \lambda(u_0u_1) \cdots \lambda(u_{q-1}u_q)$.

Oracle: Let $U = D$ and $\epsilon = 1_\Gamma$. We define $\mathcal{A}(a, e) = a \cdot \lambda(e)$ and $\mathcal{T}(a, b) = [a = b]$. Both the functions take $O(T_\Gamma)$ time.

LP for SUBSET FEEDBACK VERTEX SET

Input: A graph $G = (V, E)$, a set $S \subseteq V$, and a root $s \in V$.

\mathcal{F} : $C \in \mathcal{F}$ iff $\mathbf{1}_{E(C)}(e)$ is odd for some edge e incident to S .

(\equiv) : $P \equiv Q$ iff $\mathbf{1}_{E(P)}(e) \equiv \mathbf{1}_{E(Q)}(e) \pmod{2}$ for all e incident to S

Oracle: Observe that for a single branching pair (P, Q) , $P \equiv Q$ if and only if the simple cycle contained in $P \circ Q^{-1}$ does not contain any vertex in S . Let $U = E \cup \{\epsilon\}$, which will represent the last passed edge incident to S . The append function is defined as $\mathcal{A}(a, e) = e$ if e is incident to S and otherwise $\mathcal{A}(a, e) = a$. The test function is defined as $\mathcal{T}(a, b) = [a = b]$. Both the functions take a constant time.

LP for NODE MULTIWAY CUT

Input: A graph $G = (V, E)$, a root $s \in V$, and a partition $\{\{t_0, \dots, t_k\} \mid t \in T\}$ of the neighbors of s .

\mathcal{F} : $C \in \mathcal{F}$ iff $\sum_{i=0}^k \mathbf{1}_{E(C)}(st_i)$ is odd for some terminal $t \in T$.

(\equiv) : $P \equiv Q$ iff $\sum_{i=0}^k \mathbf{1}_{E(P)}(st_i) \equiv \sum_{i=0}^k \mathbf{1}_{E(Q)}(st_i) \pmod{2}$ for all terminal $t \in T$.

Oracle: Let $U = T \cup \{\epsilon\}$. The append function is defined as $\mathcal{A}(\epsilon, st_i) = t$, $\mathcal{A}(t, t_i s) = \epsilon$, and otherwise $\mathcal{A}(a, e) = a$. The test function is defined as $\mathcal{T}(a, b) = [a = b]$. Both the functions take a constant time.

LP for NODE UNIQUE LABEL COVER

Input: A finite alphabet Σ , a graph $G = (V, E)$, a permutation π_e of Σ for every edge $e \in \hat{E}$ such that $\pi_{uv} = \pi_{vu}^{-1}$, a root $s \in V$, and an element $d \in \Sigma$.

\mathcal{F} : $C \in \mathcal{F}$ iff $\pi_C^*(d) \neq d$, where $\pi_e^*(a) := \pi_e(a)$ and $\pi_{P \circ e}^*(a) := \pi_e(\pi_P^*(a))$.

(\equiv) : $P \equiv Q$ iff $\pi_P(d) = \pi_Q(d)$.

Oracle: let $U = \Sigma$ and $\epsilon = d$. The append function is defined as $\mathcal{A}(a, e) = \pi_e(a)$. The test function is defined as $\mathcal{T}(a, b) = [a = b]$. Both the functions take a constant time.

LP for NON-MONOCROMATIC CYCLE TRANSVERSAL

Input: An edge-colored graph $G = (V, E)$ and a root $s \in V$.

\mathcal{F} : Let $\#(W, v, c) := \sum_{e \in \delta(v): e \text{ has color } c} \mathbf{1}_{E(W)}(e)$. $C \in \mathcal{F}$ iff there exists a vertex $v \in V$ and a color c such that $\#(C, v, c)$ is odd.

(\equiv) : $P \equiv Q$ iff for any vertex v and any color c , $\#(P, v, c) \equiv \#(Q, v, c) \pmod{2}$

Oracle: Observe that for a single branching pair (P, Q) , $P \equiv Q$ if and only if the simple cycle contained in $P \circ Q^{-1}$ is monochromatic. Let C be the set of colors and let $U = (V \times C) \cup \{\epsilon, *\}$. An element $(w, c) \in U$ represents that all the edges in the subpath from the vertex w are colored by c , and the element $*$ $\in U$ represents that the s -path induces a monochromatic cycle. Let c be the color of an edge e from u to v . The append function is defined as $\mathcal{A}(\epsilon, e) = (s, c)$, $\mathcal{A}((v, c), e) = *$, $\mathcal{A}((w, c), e) = (w, c)$, and $\mathcal{A}((w, c'), e) = (u, c)$ for $c' \neq c$. The test function is defined as $\mathcal{T}(a, *) = \mathcal{T}(*, b) = 1$ and otherwise $\mathcal{T}(a, b) = [a = b]$. Both the functions take a constant time.

Note that even if \mathcal{F} is nice, the test function may have a tuple $a, b, c \in U$ such that $\mathcal{T}(a, b) = \mathcal{T}(b, c) = 1$ but $\mathcal{T}(a, c) = 0$ as shown in the example for NON-MONOCROMATIC CYCLE TRANSVERSAL.

4.2 Finding Half-Integral Farthest Minimum \mathcal{F} -Cover

For an \mathcal{F} -cover x , let $R(x)$ denote the set of vertices reachable from s by using only vertices of value zero in x . A minimum \mathcal{F} -cover x is called *farthest* if there exists no minimum \mathcal{F} -cover x' such that $R(x) \subset R(x')$. The following is a direct implication of the results in [18, 35] and the constructions of the equivalence oracle shown in Section 4.1.

Theorem 2 ([18, 35]). *If a half-integral farthest minimum \mathcal{F} -cover of size k can be computed in $T(k, m, T_{\mathcal{E}})$ time for every graph of m edges and every nice set \mathcal{F} given as an equivalence oracle implemented by $T_{\mathcal{E}}$ -time append/test functions, we obtain the following results.*

- GROUP FEEDBACK VERTEX SET can be solved in $O(4^k T(k, m, T_{\Gamma}))$ time.
- SUBSET FEEDBACK VERTEX SET can be solved in $O(4^k T(k, m, 1))$ time.
- NODE MULTIWAY CUT can be solved in $O(2^k T(k, m, 1))$ time.
- NODE UNIQUE LABEL COVER can be solved in $O(|\Gamma|^{2k} T(k, m, 1))$ time.
- NON-MONOCROMATIC CYCLE TRANSVERSAL can be solved in $O(4^k T(k, m, 1))$ time.

In this section, we give an algorithm running in $T(k, m, T_{\mathcal{E}}) = O(kmT_{\mathcal{E}})$ time.

Theorem 3. *Given a graph of m edges and an equivalence oracle of a nice set \mathcal{F} implemented by $T_{\mathcal{E}}$ -time append/test functions, a half-integral farthest minimum \mathcal{F} -cover of size k can be computed in $O(kmT_{\mathcal{E}})$ time.*

Thus, we immediately obtain the following corollary.

Corollary 4. *The following holds.*

- GROUP FEEDBACK VERTEX SET can be solved in $O(4^k kmT_{\Gamma})$ time.
- SUBSET FEEDBACK VERTEX SET can be solved in $O(4^k km)$ time.
- NODE MULTIWAY CUT can be solved in $O(2^k km)$ time.
- NODE UNIQUE LABEL COVER can be solved in $O(|\Gamma|^{2k} km)$ time.
- NON-MONOCROMATIC CYCLE TRANSVERSAL can be solved in $O(4^k km)$ time.

Note that for each of these problems, the base of the exponent coincides with the current best one. For obtaining the algorithm, we first prove two lemmas.

Lemma 13. *For any basic maximum \mathcal{F} -packing y and any minimum \mathcal{F} -cover x , the following holds.*

1. $x(V(C)) = 1$ for any integral cycle C of y .

2. $x(V(S)) = \frac{1}{2}$ for any spoke S of y .

Proof. For a wheel W , we denote the degree of W by $d(W)$ and the set of vertices contained in W by $V(W)$. Note that $V(W)$ is not a multiset. First, we prove that $x(V(W))$ is always at least $\frac{d(W)}{2}$ and if the equality holds, then $x(V(S)) = \frac{1}{2}$ holds for any spoke S of W . Let W be a wheel with a half-integral cycle $H_1 \circ \dots \circ H_d$ and spokes $\{S_1, \dots, S_d\}$. We define $H := V(W) \setminus (V(S_1) \cup \dots \cup V(S_d))$. Then, we have

$$\begin{aligned} x(V(W)) &= x(H) + \sum_{i=1}^d x(V(S_i)) \\ &= \frac{1}{2}x(H) + \frac{1}{2} \sum_{i=1}^d x(V(S_i \circ H_i \circ S_{i+1}^{-1})) \\ &\geq \frac{1}{2} \sum_{i=1}^d 1 \\ &= \frac{d}{2}. \end{aligned}$$

If $x(V(W)) = \frac{d}{2}$ holds, we have $x(H) = 0$ and $x(V(S_i)) + x(V(S_{i+1})) = 1$ for any $i \in \{1, \dots, d\}$. Because d is odd, this implies that $x(V(S_i)) = \frac{1}{2}$ holds for any $i \in \{1, \dots, d\}$.

Now, we prove the lemma. Because all the integral cycles and wheels do not share any vertices other than the root s , we have

$$\begin{aligned} |x| &\geq \sum_{C:\text{integral cycle}} x(V(C)) + \sum_{W:\text{wheel}} x(V(W)) \\ &\geq \sum_{C:\text{integral cycle}} 1 + \sum_{W:\text{wheel}} \frac{d(W)}{2} \\ &= |y| \\ &= |x|. \end{aligned}$$

Therefore, $x(V(C)) = 1$ and $x(V(W)) = \frac{d(W)}{2}$ must hold for any integral cycle C and any wheel W . \square

Lemma 14. For a graph G , a nice set \mathcal{F} , and a path P from the root s to a vertex t , let G' be a graph obtained by inserting an edge $e = st$ with $e \equiv P$ (i.e., \mathcal{F} is extended so that $e \circ Q^{-1} \in \mathcal{F}$ iff $P \circ Q^{-1} \in \mathcal{F}$ for any walk Q from s to t)¹⁷. Then, any \mathcal{F} -cover x of G satisfying $x(V(P) \setminus \{t\}) = 0$ is also an \mathcal{F} -cover of G' .

Proof. Suppose that there exists an s -cycle $C \in \mathcal{F}$ of G' with $x(V(C)) < 1$. Because x is an \mathcal{F} -cover of G , C must contain the edge e . Therefore, we can write $C = e \circ Q^{-1}$ for some s -walk $Q \not\equiv e$. Then, we have $x(V(P \circ Q^{-1})) = x(V(C)) < 1$ and $P \equiv e \not\equiv Q$, which contradicts the fact that x is an \mathcal{F} -cover of G . \square

¹⁷Technically, this is equivalent to contracting $V(P) \setminus \{t\}$ to s . We use the edge insertion instead of the contraction for avoiding a discussion about how to obtain an oracle for the contracted graph.

Algorithm 2 Algorithm for computing a half-integral farthest minimum \mathcal{F} -cover

```
1: Compute a basic maximum  $\mathcal{F}$ -packing  $y$ .
2: for each integral cycle  $C = (v_0, \dots, v_l)$  of  $y$  do
3:   while  $a(C) < b(C)$  do
4:     Insert an edge  $e = sv_{a(C)+1}$  with  $e \equiv (v_0, \dots, v_{a(C)+1})$ .
5:     if there exists a  $y$ -augmenting path or pair then
6:       Remove  $e$ . break
7:   while  $a(C) < b(C)$  do
8:     Insert an edge  $e = sv_{b(C)-1}$  with  $e \equiv (v_l, \dots, v_{b(C)-1})$ .
9:     if there exists a  $y$ -augmenting path or pair then
10:      Remove  $e$ . break
11: for each spoke  $S = (v_0, \dots, v_l)$  of  $y$  do
12:   while  $a(S) < l$  do
13:     Insert an edge  $e = sv_{a(S)+1}$  with  $e \equiv (v_0, \dots, v_{a(S)+1})$ .
14:     if there exists a  $y$ -augmenting path or pair then
15:       Remove  $e$ . break
16: Construct the corresponding half-integral minimum  $\mathcal{F}$ -cover  $x$ .
17: return  $x$ 
```

Now, we describe the algorithm for computing a half-integral farthest minimum \mathcal{F} -cover (see Algorithm 2). In the algorithm, we iteratively modify the graph by inserting edges. We denote the current graph by G and the original graph by G_{orig} . First, we compute a basic maximum \mathcal{F} -packing y by using the algorithm in Section 3. We keep and reuse the tables (a , b , prev, and tail) used in the last execution of Algorithm 1 which returned NO. We process the integral cycles and the spokes of y one by one, whose detail is described later, by preserving the following invariants.

Lemma 15. *The following invariants hold at any step of Algorithm 2.*

1. y is a maximum \mathcal{F} -packing of G .
2. Any \mathcal{F} -cover x of G_{orig} satisfying the following conditions is also an \mathcal{F} -cover of G .
 - (a) For any integral cycle $C = (v_0, \dots, v_l)$, $x(v_i) = 0$ holds for all $i \in \{0, \dots, a(C) - 1\} \cup \{b(C) + 1, \dots, l\}$.
 - (b) For any spoke $S = (v_0, \dots, v_l)$, $x(v_i) = 0$ holds for all $i \in \{0, \dots, a(S) - 1\}$.
3. For any processed integral cycle $C = (v_0, \dots, v_l)$, there exists no minimum \mathcal{F} -cover x of G satisfying $x(v_i) = 0$ for all $i \in \{0, \dots, a(C)\} \cup \{b(C) + 1, \dots, l\}$ or all $i \in \{0, \dots, a(C) - 1\} \cup \{b(C), \dots, l\}$.
4. For any processed spoke $S = (v_0, \dots, v_l)$, there exists no minimum \mathcal{F} -cover x of G satisfying $x(v_i) = 0$ for all $i \in \{0, \dots, a(S)\}$.

When all the integral cycles and the spokes are processed, we return the half-integral minimum \mathcal{F} -cover of G constructed from the current tables a and b as described in Section 3.3. From these invariants, we can easily prove the correctness of the algorithm.

Lemma 16. *When all the integral cycles and the spokes are processed, the half-integral minimum \mathcal{F} -cover x of G constructed from the current tables a and b is a farthest minimum \mathcal{F} -cover of G_{orig} .*

Proof. From the invariants 3 or 4, x is a farthest minimum \mathcal{F} -cover of G . Because G_{orig} is a subgraph of G , x is also an \mathcal{F} -cover of G_{orig} , and moreover, from the invariant 1, x is a minimum \mathcal{F} -cover of G_{orig} . Suppose that x is not a farthest minimum \mathcal{F} -cover of G_{orig} . Then, there exists a minimum \mathcal{F} -cover x' of G_{orig} such that $R(x) \subset R(x')$. From the construction of x , x' satisfies the conditions in the invariant 2, and therefore, x' is a minimum \mathcal{F} -cover of G , which contradicts the fact that x is a farthest minimum \mathcal{F} -cover of G . \square

Now, we describe how to process the integral cycles and the spokes. For each integral cycle $C = (v_0, \dots, v_l)$, we first repeat the following while $a(C) < b(C)$ holds (lines 3–6). Let $i := a(C) + 1$. We insert an edge $e = sv_i$ with $e \equiv (v_0, \dots, v_i)$, and then we search for a y -augmenting path or pair by using Algorithm 1. Here, instead of searching for a y -augmenting path/pair from scratch by initializing the tables, we restart the search from line 4 of Algorithm 1 by setting $X \leftarrow \{v_{a(C)}\}$ and reusing the current tables updated as $\text{prev}(v_{a(C)}) \leftarrow e$, $\text{tail}(v_{a(C)}) = \mathcal{A}^*((v_0, \dots, v_{a(C)}))$, and $a(C) \leftarrow a(C) + 1$. Note that because the root s has been already processed, the algorithm never calls the append function \mathcal{A} against the inserted edge e . Therefore, we do not have to update the equivalence oracle. If the restarted search finds a y -augmenting path or pair, we remove e , rewind all the changes in this step (we do not rewind the changes in the previous steps where the restarted searches returned NO), and exit the repetition. If the restarted search returns NO, we keep the tables updated by the restarted search and continue the repetition.

Claim 5. *The restarted search can correctly compute a y -augmenting path or pair if exists.*

Proof. We can virtually think as follows. We first subdivide the edge $e = sv_i$ by introducing a vertex w and then run Algorithm 1 from scratch. Because the order of vertices picked at line 5 of Algorithm 1 is arbitrary, we can choose completely the same order as the one in the last execution that has constructed the current tables before the update. Note that, from the invariant 3b in Lemma 4, v_i was not visited in the last execution, and therefore, the insertion of e does not affect the search at all. In the end, the execution reaches to a condition such that $X = \{w\}$ and all the tables are completely same as the current tables before the update. Then the algorithm picks the vertex w and the edge to v_i . Because $e \equiv (v_0, \dots, v_i)$ holds, we have $e \not\equiv (v_l, \dots, v_i)$. Therefore, the lines 14–17 of Algorithm 1 are executed. Now, $v_{a(C)}$ is inserted into X and the tables are updated as $\text{prev}(v_{a(C)}) \leftarrow e$, $\text{tail}(v_{a(C)}) = \mathcal{A}^*((v_0, \dots, v_{a(C)}))$, and $a(C) \leftarrow a(C) + 1$, which are completely same as the current tables after the update. Thus, instead of running Algorithm 1 from scratch, we can use the restarted search. \square

Claim 6. *Lines 3–6 preserve all the invariants.*

Proof. Because we keep the changes only when we fail to find a y -augmenting path or pair, the invariant 1 is satisfied. The invariant 2 follows from Lemma 14 against $P := (v_0, \dots, v_i)$. The invariants 3 and 4 are trivial. Note that the current integral cycle C is still under processing. \square

Claim 7. *When the repetition of lines 3–6 ends, there exists no minimum \mathcal{F} -cover x of G satisfying $x(v_i) = 0$ for all $i \in \{0, \dots, a(C)\} \cup \{b(C) + 1, \dots, l\}$.*

Proof. The repetition ends when $a(C)$ becomes equal to $b(C)$ or when a y -augmenting path or pair is found. In the former case, we have $\{0, \dots, a(C)\} \cup \{b(C) + 1, \dots, l\} = \{0, \dots, l\}$. Therefore, from Lemma 13, there exists no minimum \mathcal{F} -cover of G satisfying the condition in the lemma. In the latter case, let G' be the graph obtained from G by inserting the edge $sv_{a(C)+1}$. Suppose that there exists a minimum \mathcal{F} -cover x of G satisfying the condition in the lemma. Then, from Lemma 14, x is also an \mathcal{F} -cover of G' . On the other hand, because G' has a y -augmenting path or pair, the size of the maximum \mathcal{F} -packing y' of G' is strictly larger than $|y|$. Thus, we have $|y'| > |y| = |x| \geq |y'|$, which is a contradiction. \square

After repeating lines 3–6, we repeat the following while $a(C) < b(C)$ holds (lines 7–10). Let $i := b(C) - 1$. We insert an edge $e = sv_i$ with $e \equiv (v_l, \dots, v_i)$, and then we restart the search by setting $X \leftarrow \{v_{b(C)}\}$ and reusing the current tables updated as $\text{prev}(v_{b(C)}) \leftarrow e$, $\text{tail}(v_{b(C)}) = \mathcal{A}^*((v_l, \dots, v_{b(C)}))$, and $b(C) \leftarrow b(C) - 1$. If the restarted search finds a y -augmenting path or pair, we remove e , rewind all the changes in this step, and exit the repetition. If the restarted search returns NO, we keep the tables updated by the restarted search and continue the repetition. By the same argument as in the case of lines 3–6, all the invariants are preserved and, when the repetition ends, there exists no minimum \mathcal{F} -cover x of G satisfying $x(v_i) = 0$ for all $i \in \{0, \dots, a(C) - 1\} \cup \{b(C), \dots, l\}$. Thus, the invariant 3 is satisfied for C when we have finished processing C .

Next, for each spoke $S = (v_0, \dots, v_l)$, we repeat the following while $a(S) < l$ holds (lines 12–15). Let $i := a(S) + 1$. We insert an edge $e = sv_i$ with $e \equiv (v_0, \dots, v_i)$, and then we restart the search by setting $X \leftarrow \{v_{a(S)}\}$ and reusing the current tables updated as $\text{prev}(v_{a(S)}) \leftarrow e$, $\text{tail}(v_{a(S)}) = \mathcal{A}^*((v_0, \dots, v_{a(S)}))$, and $a(S) \leftarrow a(S) + 1$. If the restarted search finds a y -augmenting path or pair, we remove e , rewind all the changes in this step, and exit the repetition. If the restarted search returns NO, we keep the tables updated by the restarted search and continue the repetition. By the same argument as in the case of integral cycles, all the invariants are preserved and, when the repetition ends, there exists no minimum \mathcal{F} -cover x of G satisfying $x(v_i) = 0$ for all $i \in \{0, \dots, a(S)\}$. Thus, the invariant 4 is satisfied for S when we have finished processing S .

Finally, we analyze the running time of Algorithm 2. The number of inserted edges is $O(n)$ and the number of while loops (lines 3–6, 7–10, or 12–15) is $2k$. For each while loop, a series of the restarted searches can be regarded as a single execution of Algorithm 1 which takes $O(mT_{\mathcal{E}})$ time. Therefore, the total running time is $O(kmT_{\mathcal{E}})$. Thus, we obtain Theorem 3.

References

- [1] M. A. Babenko. A fast algorithm for the path 2-packing problem. *Theory of Computing Systems*, 46(1):59, 2010.
- [2] M. L. Balinski. Integer programming: Methods, uses, computations. *Management Science*, 12(3):253–313, 1965.
- [3] A. Becker, R. Bar-Yehuda, and D. Geiger. Randomized algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research*, 12:219–234, 2000.
- [4] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 2006.

- [5] J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.
- [6] M. Chudnovsky, W. H. Cunningham, and J. Geelen. An algorithm for packing non-zero A -paths in group-labelled graphs. *Combinatorica*, 28(2):145–161, 2008.
- [7] M. Chudnovsky, J. Geelen, B. Gerards, L. Goddyn, M. Lohman, and P. Seymour. Packing non-zero A -paths in group-labelled graphs. *Combinatorica*, 26(5):521–532, 2006.
- [8] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.
- [9] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Transactions on Computation Theory*, 5(1):3–11, 2013.
- [10] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer Verlag, 2012.
- [11] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, pages 449–467, 1965.
- [12] A. Frank and É. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.
- [13] S. Fujishige and X. Zhang. New algorithms for the intersection problem of submodular systems. *Japan Journal of Industrial and Applied Mathematics*, 9(3):369, 1992.
- [14] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49–61, 2004.
- [15] S. Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011.
- [16] H. Hirai. A dual descent algorithm for node-capacitated multiflow problems and its applications. 2015. [arXiv:1508.07065](#).
- [17] Y. Iwata, K. Oka, and Y. Yoshida. Linear-time FPT algorithms via network flow. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1749–1761, 2014.
- [18] Y. Iwata, M. Wahlström, and Y. Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM Journal on Computing*, 45(4):1377–1411, 2016.
- [19] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014.
- [20] D. Lokshtanov, M. S. Ramanujan, and S. Saurabh. Linear time parameterized algorithms for subset feedback vertex set. In *Proceedings of 42nd International Colloquium on the Automata, Languages, and Programming (ICALP)*, pages 935–946, 2015.
- [21] D. Lokshtanov, M. S. Ramanujan, and S. Saurabh. A Linear Time Parameterized Algorithm for Directed Feedback Vertex Set. 2016. [arXiv:1609.04347](#).

- [22] D. Lokshtanov, M. S. Ramanujan, and S. Saurabh. A linear time parameterized algorithm for node unique label cover. 2016. [arXiv:1604.08764](#).
- [23] L. Lovász. The matroid matching problem. *Algebraic Methods in Graph Theory*, 1:495–517, 1978.
- [24] L. Lovász. Matroid matching and some applications. *Journal of Combinatorial Theory, Series B*, 28(2):208–236, 1980.
- [25] W. Mader. Über die Maximalzahl kreuzungsfreier H -Wege. *Archiv der Mathematik*, 31(1):387–402, 1978.
- [26] G. Nemhauser and L. Trotter. Vertex packing: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.
- [27] G. Pap. *A constructive approach to matching and its generalizations*. PhD thesis, Ph. D. thesis, Eötvös Loránd University, 2006.
- [28] G. Pap. Packing non-returning A -paths. *Combinatorica*, 27(2):247–251, 2007.
- [29] G. Pap. Some new results on node-capacitated packing of A -paths. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 599–604, 2007.
- [30] G. Pap. Packing non-returning A -paths algorithmically. *Discrete Mathematics*, 308(8):1472–1488, 2008.
- [31] G. Pap. Strongly polynomial time solvability of integral and half-integral node-capacitated multiflow problems. Technical report, Citeseer, 2008.
- [32] M. S. Ramanujan and S. Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1739–1748, 2014.
- [33] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.
- [34] S. Tanigawa and Y. Yamaguchi. Packing non-zero A -paths via matroid matching. *Discrete Applied Mathematics*, 214:169–178, 2016.
- [35] M. Wahlström. LP-branching algorithms based on biased graphs. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1559–1570, 2017.
- [36] Y. Yamaguchi. Packing A -paths in group-labelled graphs via linear matroid parity. *SIAM Journal on Discrete Mathematics*, 30(1):474–492, 2016.

A List of Problems

We give definitions of the problems considered in this paper.

FEEDBACK VERTEX SET

Input: A graph $G = (V, E)$ and an integer k .

Question: Is there a set $X \subseteq V$ of at most k vertices such that $G - X$ is a forest?

GROUP FEEDBACK VERTEX SET

Input: A group $\Gamma = (D, \cdot)$ given as an $O(T_\Gamma)$ -time oracle performing the group operation (\cdot) , a Γ -labeled graph $G = (V, E)$ with labeling $\lambda : \hat{E} \rightarrow D$ with $\lambda(uv) \cdot \lambda(vu) = 1_\Gamma$ for every $uv \in \hat{E}$, where 1_Γ is the unity of Γ , and an integer k .

Question: Is there a set $X \subseteq V$ of at most k vertices such that $G - X$ has a consistent labeling? That is, is there a labeling $\varphi : V \setminus X \rightarrow D$ such that $\varphi(u) \cdot \lambda(uv) = \varphi(v)$ for every $uv \in E(G - X)$?

SUBSET FEEDBACK VERTEX SET

Input: A graph $G = (V, E)$, a set $S \subseteq V$, and an integer k .

Question: Is there a set $X \subseteq V$ of at most k vertices such that there is no cycle passing through a vertex of S in $G - X$?

NODE MULTIWAY CUT

Input: A graph $G = (V, E)$, a set of terminals $T \subseteq V$, and an integer k .

Question: Is there a set $X \subseteq V \setminus T$ of size at most k such that every terminal in T lies in a different connected component of $G - X$?

NODE UNIQUE LABEL COVER

Input: A finite alphabet Σ , a graph $G = (V, E)$, a permutation π_e of Σ for every edge $e \in \hat{E}$ such that $\pi_{uv} = \pi_{vu}^{-1}$, and an integer k .

Question: Is there a set $X \subseteq V$ of at most k vertices and a function $\varphi : V \setminus X \rightarrow \Sigma$ such that $\pi_{uv}(\varphi(u)) = \varphi(v)$ for every $uv \in E(G - X)$?

NON-MONOCROMATIC CYCLE TRANSVERSAL

Input: An edge-colored graph $G = (V, E)$ and an integer k .

Question: Is there a set $X \subseteq V$ of at most k vertices such that $G - X$ contains no non-monochromatic cycles?